* # Load the datasets
* df\_fake = pd.read\_csv("Fake.csv.zip")
* df\_real = pd.read\_csv("True.csv.zip")
* # Check top rows
* print("Fake News:")
* print(df\_fake.head())
* print("\nReal News:")
* print(df\_real.head())
* import pandas as pd
* # Load dataset (replace with your filename or URL if needed)
* df = pd.read\_csv("Fake.csv.zip")  # or "True.csv"
* # Preview original data
* print("Original Dataset Info:")
* print(df.info())
* # Check for missing values
* print("\nMissing Values Per Column:")
* print(df.isnull().sum())
* # Option 1: Drop rows with any missing values
* df\_cleaned = df.dropna()
* # Option 2: (Alternative) Fill missing values
* # df\_filled = df.fillna("Unknown")  # Or a specific value
* # After cleaning
* print("\nCleaned Dataset Info:")
* print(df\_cleaned.info())
* # Optional: Save cleaned file
* # df\_cleaned.to\_csv("Fake\_cleaned.csv", index=False)
* #Load dataset (replace with your filename or URL if needed)
* df = pd.read\_csv("True.csv.zip")  # or "True.csv"
* # Preview original data
* print("Original Dataset Info:")
* print(df.info())
* # Check for missing values
* print("\nMissing Values Per Column:")
* print(df.isnull().sum())
* # Option 1: Drop rows with any missing values
* df\_cleaned = df.dropna()
* # Option 2: (Alternative) Fill missing values
* # df\_filled = df.fillna("Unknown")  # Or a specific value
* # After cleaning
* print("\nCleaned Dataset Info:")
* print(df\_cleaned.info())
* # Optional: Save cleaned file
* # df\_cleaned.to\_csv("Fake\_cleaned.csv", index=False)
* import pandas as pd
* importmatplotlib.pyplotasplt
* importseabornassns
* # Load the dataset
* df\_fake = pd.read\_csv("Fake.csv.zip")
* df\_real = pd.read\_csv("True.csv.zip")
* # Label the data
* df\_fake['label'] = 0
* df\_real['label'] = 1
* # Combine the data
* df = pd.concat([df\_fake,df\_real], axis=0).reset\_index(drop=True)
* # Basic info
* print("Dataset Info:")
* print(df.info())
* # Check for missing values
* print("\nMissing values:")
* print(df.isnull().sum())
* # Dataset shape
* print(f"\nDataset Shape: {df.shape}")
* # Label distribution
* plt.figure(figsize=(6,4))
* sns.countplot(data=df, x='label')
* plt.title("Distribution of Real (1) and Fake (0) News")
* plt.xlabel("Label")
* plt.ylabel("Count")
* plt.xticks([0,1],['Fake','Real'])
* plt.show()
* # Word count distribution
* df['text\_length'] = df['text'].astype(str).apply(lambda x:len(x.split()))
* plt.figure(figsize=(10,5))
* sns.histplot(data=df, x='text\_length', hue='label', bins=50,kde=True)
* plt.title("Text Length Distribution by Label")
* plt.xlabel("Number of Words in Article")
* plt.ylabel("Frequency")
* plt.legend(title='Label', labels=['Fake','Real'])
* plt.show()
* # Most frequent words in fake news
* fromsklearn.feature\_extraction.textimportCountVectorizer
* cv = CountVectorizer(stop\_words='english',max\_features=20)
* X\_fake = cv.fit\_transform(df[df['label'] == 0]['text'].astype(str))
* words\_fake = cv.get\_feature\_names\_out()
* counts\_fake = X\_fake.toarray().sum(axis=0)
* plt.figure(figsize=(12,6))
* sns.barplot(x=counts\_fake, y=words\_fake)
* plt.title("Top 20 Words in Fake News")
* plt.xlabel("Frequency")
* plt.ylabel("Word")
* plt.show()
* import pandas as pd
* import re
* fromsklearn.model\_selectionimporttrain\_test\_split
* fromsklearn.feature\_extraction.textimportTfidfVectorizer
* fromsklearn.linear\_modelimportLogisticRegression
* fromsklearn.metricsimportaccuracy\_score,classification\_report,confusion\_matrix
* importmatplotlib.pyplotasplt
* importseabornassns
* # Load and label the data
* df\_fake = pd.read\_csv("Fake.csv.zip")
* df\_real = pd.read\_csv("True.csv.zip")
* df\_fake['label'] = 0  # Fake
* df\_real['label'] = 1  # Real
* df = pd.concat([df\_fake,df\_real]).reset\_index(drop=True)
* df['text'] = df['text'].fillna("")
* # Text cleaning
* defclean\_text(text):
* text = re.sub(r'[^\w\s]','',text.lower())
* return text
* df['clean\_text'] = df['text'].apply(clean\_text)
* # Split features and labels
* X = df['clean\_text']
* y = df['label']
* # Train-test split
* X\_train,X\_test,y\_train,y\_test = train\_test\_split(X, y,test\_size=0.2,random\_state=42)
* # Vectorization
* tfidf = TfidfVectorizer(stop\_words='english',max\_df=0.7)
* X\_train\_tfidf = tfidf.fit\_transform(X\_train)
* X\_test\_tfidf = tfidf.transform(X\_test)
* # Build model
* model = LogisticRegression(max\_iter=1000)
* model.fit(X\_train\_tfidf,y\_train)
* # Predictions
* y\_pred = model.predict(X\_test\_tfidf)
* # Evaluation
* print("Accuracy:",accuracy\_score(y\_test,y\_pred))
* print("\nClassification Report:")
* print(classification\_report(y\_test,y\_pred))
* # Confusion Matrix
* cm = confusion\_matrix(y\_test,y\_pred)
* plt.figure(figsize=(6,4))
* sns.heatmap(cm,annot=True,fmt="d",cmap="Blues",xticklabels=["Fake","Real"],yticklabels=["Fake","Real"])
* plt.xlabel("Predicted")
* plt.ylabel("Actual")
* plt.title("Confusion Matrix")
* plt.show()
* fromsklearn.metricsimport(
* accuracy\_score,
* precision\_score,
* recall\_score,
* f1\_score,
* classification\_report,
* confusion\_matrix,
* roc\_auc\_score,
* roc\_curve
* )
* importmatplotlib.pyplotasplt
* importseabornassns
* # Assuming y\_test and y\_pred are already defined
* # Also assuming y\_proba if using ROC-AUC
* # Accuracy
* accuracy = accuracy\_score(y\_test,y\_pred)
* print(f"Accuracy: {accuracy:.4f}")
* # Precision, Recall, F1-score
* precision = precision\_score(y\_test,y\_pred)
* recall = recall\_score(y\_test,y\_pred)
* f1 = f1\_score(y\_test,y\_pred)
* print(f"Precision: {precision:.4f}")
* print(f"Recall:    {recall:.4f}")
* print(f"F1 Score:  {f1:.4f}")
* # Full classification report
* print("\nClassification Report:")
* print(classification\_report(y\_test,y\_pred,target\_names=["Fake","Real"]))
* # Confusion Matrix
* cm = confusion\_matrix(y\_test,y\_pred)
* plt.figure(figsize=(6,4))
* sns.heatmap(cm,annot=True,fmt="d",cmap="Purples",xticklabels=["Fake","Real"],yticklabels=["Fake","Real"])
* plt.title("Confusion Matrix")
* plt.xlabel("Predicted Label")
* plt.ylabel("True Label")
* plt.show()
* # ROC Curve & AUC
* # For ROC-AUC, you need predicted probabilities
* # Example for Logistic Regression: y\_proba = model.predict\_proba(X\_test\_tfidf)[:, 1]
* y\_proba = model.predict\_proba(X\_test\_tfidf)[:,1]
* fpr,tpr, thresholds = roc\_curve(y\_test,y\_proba)
* auc\_score = roc\_auc\_score(y\_test,y\_proba)
* plt.figure(figsize=(6,4))
* plt.plot(fpr,tpr, label=f"AUC = {auc\_score:.4f}")
* plt.plot([0,1],[0,1],'k--')
* plt.title("ROC Curve")
* plt.xlabel("False Positive Rate")
* plt.ylabel("True Positive Rate")
* plt.legend()
* plt.grid(True)
* plt.show()
* !pip install gradioscikit-learn pandas
* import pandas as pd
* import re
* fromsklearn.feature\_extraction.textimportTfidfVectorizer
* fromsklearn.linear\_modelimportLogisticRegression
* importjoblib
* # Load and label the dataset
* df\_fake = pd.read\_csv("Fake.csv.zip")
* df\_real = pd.read\_csv("True.csv.zip")
* df\_fake['label'] = 0
* df\_real['label'] = 1
* df = pd.concat([df\_fake,df\_real]).reset\_index(drop=True)
* df['text'] = df['text'].fillna("")
* # Clean text
* defclean\_text(text):
* returnre.sub(r'[^\w\s]','',text.lower())
* df['clean\_text'] = df['text'].apply(clean\_text)
* # TF-IDF features
* tfidf = TfidfVectorizer(stop\_words='english',max\_df=0.7)
* X = tfidf.fit\_transform(df['clean\_text'])
* y = df['label']
* # Train model
* model = LogisticRegression(max\_iter=1000)
* model.fit(X, y)
* # Save model and vectorizer
* joblib.dump(model,'model.pkl')
* joblib.dump(tfidf,'vectorizer.pkl')
* importgradioas gr
* importjoblib
* import re
* # Load model and vectorizer
* model = joblib.load('model.pkl')
* vectorizer = joblib.load('vectorizer.pkl')
* defclean\_text(text):
* returnre.sub(r'[^\w\s]','',text.lower())
* # Define prediction function
* defpredict\_news(text):
* cleaned = clean\_text(text)
* vectorized = vectorizer.transform([cleaned])
* prediction = model.predict(vectorized)[0]
* return"Real News"if prediction == 1else"Fake News"
* # Gradio UI
* interface = gr.Interface(fn=predict\_news,
* inputs=gr.Textbox(lines=4, label="Enter News Article"),
* outputs=gr.Label(label="Prediction"),
* title="Fake News Detector",
* description="Enter a news article or headline. The model will predict whether it's Fake or Real.")
* # Launch app
* interface.launch()