Team Name: Data4Lyf
Team Members: **Sunayana Hazarika, JP Lai, Gaukhar Zharkeyeva, Jacob Edelson**
DubsTech Datathon 2026 Track: Technology

An Accessibility Analysis of Real-World Websites

Problem Statement: Technology is intended to be inclusive, yet many websites unintentionally exclude people with disabilities. These accessibility barriers are often invisible to designers and developers but create significant challenges for users who rely on screen readers, keyboard navigation, or who have low vision. This project analyzes real-world web accessibility violations to identify systemic patterns of exclusion in digital systems. Using the AccessGuru dataset, we examine:

1. Where accessibility failures occur most frequently
2. Which types of violations dominate, and
3. Which pages create the greatest barriers for users.

Solution Statement: This analysis prioritizes quality over complexity, combining simple analytical techniques with clear insights, actionable outcomes, and a focus on real human impact. Rather than building complex models, we emphasize interpretability, severity-based prioritization, and practical recommendations that organizations can act on immediately.

Dataset

- 3,524 accessibility violations records
- Approximately 590 unique web pages
- 6 Domains Covered: News & Media, Government, Health, Education, Technology, E-commerce
- Violations are aligned with **WCAG 2.1** standards and include both frequency and severity indicators.

Data Preparation

- Standardized domain category labels
- Removed unsuccessfully scraped pages
- Aggregated violations at page and domain levels
- Verified consistency between severity scores and impact labels

Assumptions: The dataset reflects reported violations only, so findings should be interpreted as patterns in observed accessibility failures.

Key Columns:

- domain_category: Type of website
- violation_name: What went wrong
- violation_category: Syntactic, Semantic, or Layout
- violation_score & violation_impact: How severe the issue is
- web_URL_id: Unique page identifier

Key Findings

Team Name: Data4Lyf
Team Members: **Sunayana Hazarika, JP Lai, Gaukhar Zharkeyeva, Jacob Edelson**
DubsTech Datathon 2026 Track: Technology

1. Which Domains Have the Highest Accessibility Violations?

Accessibility Failures Are Unevenly Distributed. News & Media websites exhibit the highest concentration of accessibility violations, followed by Government and Technology domains or sites.

This suggests that:

- Content-heavy sites introduce more accessibility risks
- Speed and volume of publishing may reduce accessibility checks

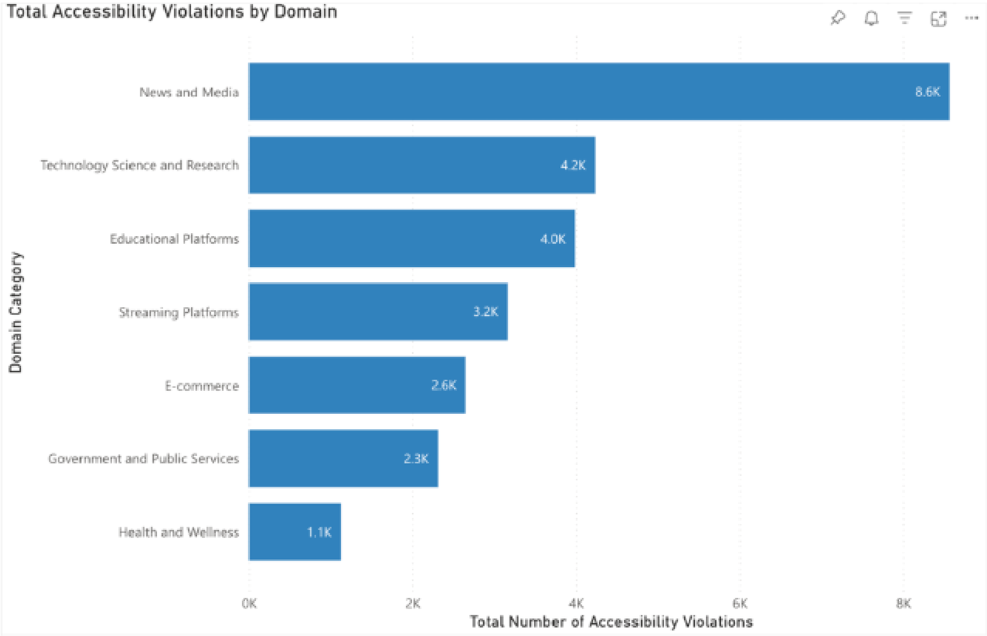


Fig1: Violations by Domain

Implications: This graph indicates that accessibility risk increases with content volume and publishing speed, making accessibility governance especially critical for these domains.

2. What Violation Types Are Most Common?

The top accessibility violations across all domains were:

- Color contrast issues
- Missing or incorrect landmarks (region)
- Missing link names
- Duplicate HTML IDs
- Heading structure problems

These violations appear repeatedly across different industries.

Team Name: Data4Lyf
Team Members: **Sunayana Hazarika, JP Lai, Gaukhar Zharkeyeva, Jacob Edelson**
DubsTech Datathon 2026 Track: Technology

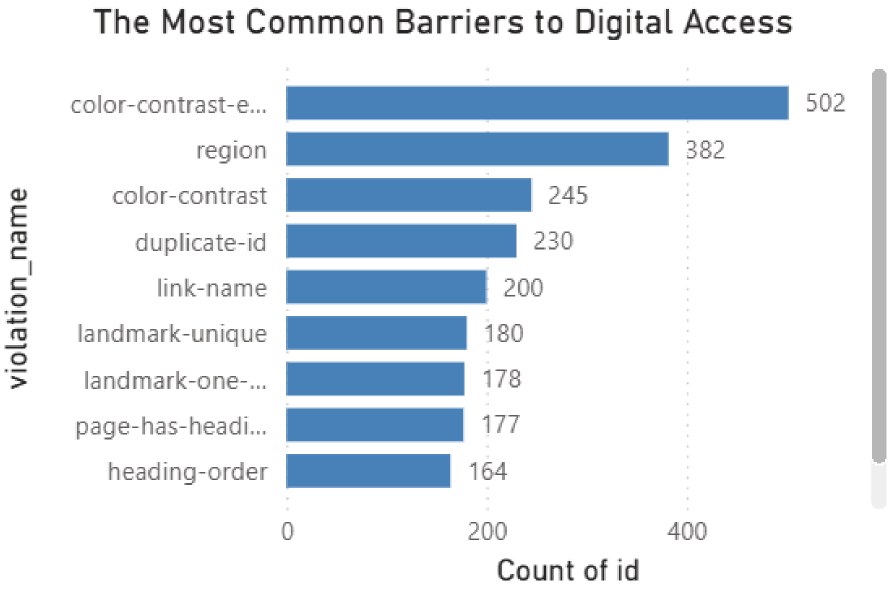
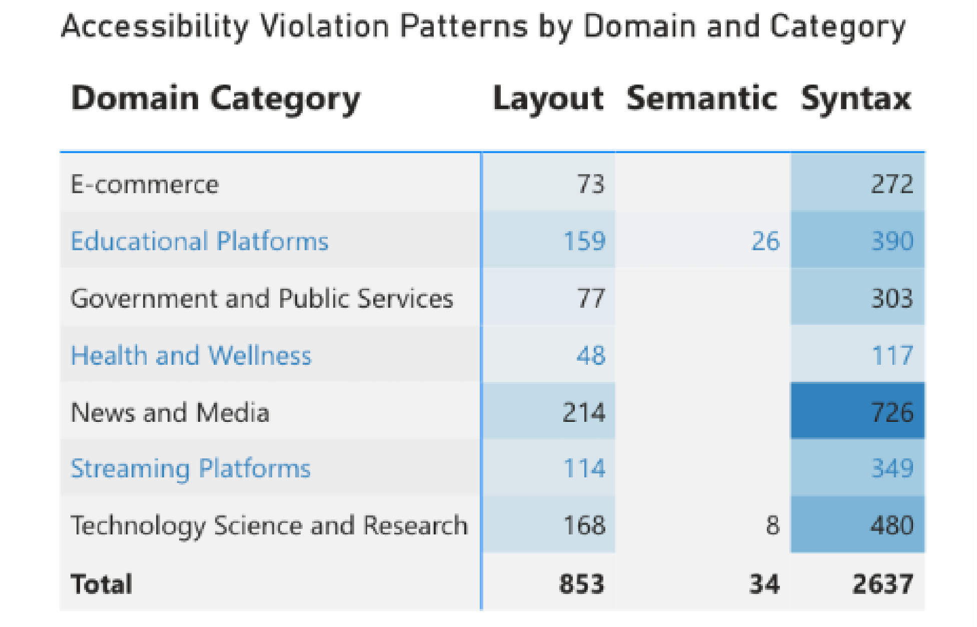


Fig 2: Top Violations

Implications: The recurrence of these violations suggests that accessibility failures are driven more by design habits and workflow gaps than by technical complexity.

3. Are There Patterns by Violation Category?



Accessibility Violation Patterns by Domain and Category

Domain Category	Layout	Semantic	Syntax
E-commerce	73		272
Educational Platforms	159	26	390
Government and Public Services	77		303
Health and Wellness	48		117
News and Media	214		726
Streaming Platforms	114		349
Technology Science and Research	168	8	480
Total	**853**	**34**	**2637**

Figure 3: Heatmap- Domain × Violation Category

Observed patterns

- Syntactic violations dominate across all domains
- Semantic violations are rarer but concentrated in Education and Technology
- Layout issues appear consistently but at lower volume

Interpretation

Most barriers arise from incorrect HTML structure and markup. Semantic violations, while fewer, disproportionately affect screen-reader users.

Severity Matters More Than Volume

Not all violations are equal. Some pages had fewer violations but more critical ones, making them much more challenging. To capture this, we introduced a severity-weighted Risk Score to identify pages that cause the greatest harm to users. A simple number that shows how harmful a page is, not just how many mistakes it has.

Risk Score Methodology

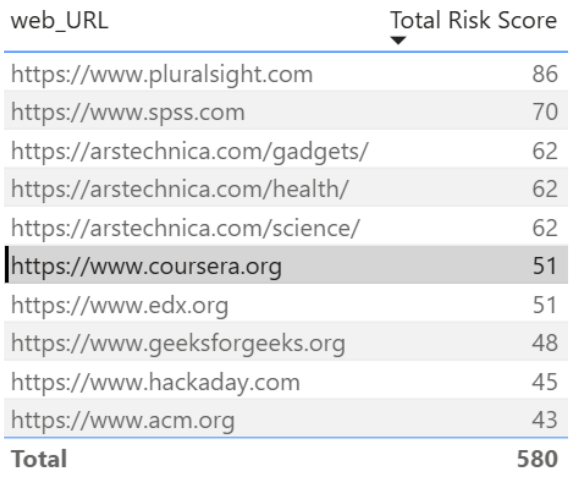
Risk Score is calculated as:

- 5 × Critical
- 4 × Serious
- 3 × Moderate
- 2 × Minor

Why does this matter?

- It highlights pages that cause real user challenges
- It helps organizations prioritize fixes
- It turns raw data into an actionable ranking

Unlike raw violation counts, the Risk Score reflects user challenges in accessing the site, allowing organizations to focus remediation efforts where they will have the greatest accessibility impact.



web_URL	Total Risk Score
https://www.pluralsight.com	86
https://www.spss.com	70
https://arstechnica.com/gadgets/	62
https://arstechnica.com/health/	62
https://arstechnica.com/science/	62
https://www.coursera.org	51
https://www.edx.org	51
https://www.geeksforgeeks.org	48
https://www.hackaday.com	45
https://www.acm.org	43
Total	**580**

Fig 4: Risk Score Ranking

Using the Risk Score, we ranked pages by accessibility harm.

Risk Score insights: Analysis of the highest-risk pages revealed that:

- A small number of pages account for a large share of critical issues
- These pages often combine:
  - Poor color contrast and
  - Missing landmarks
  - Broken page structure

Invisible Barriers and Cross-Domain Differences

Beyond visible violation counts, the analysis reveals less obvious but impactful accessibility barriers:

- Government and Public Services sites show fewer total violations but a higher proportion of structural and landmark issues, which severely affect keyboard and screen-reader navigation.
- E-commerce sites exhibit more layout-related violations tied to visual design and interactive components.
- Education and Technology domains show a higher concentration of semantic violations, impacting how assistive technologies interpret content meaning.

Interpretation:

Different industries embed different assumptions about their users. These assumptions shape distinct patterns of exclusion, often unintentionally.

The Hidden Patterns-Clustering: To identify recurring accessibility failure patterns beyond individual violations, we clustered web pages based on the types and frequency of violations they contained. We applied KMeans clustering (k = 5) on page-level violation profiles. The selected value of k balances interpretability with separation quality, achieving a silhouette score of 0.334, which indicates moderate but meaningful cluster structure in real-world accessibility data.

Method

- Per-domain representation: pivoted counts of violation types per web_URL_id
- Scaling: Standard Scaler to prevent large counts dominating similarity
- Clustering: K-Means (k = 5)
- Risk score: \text{risk} = \text{total_violations} + 2 \times \text{critical_count}
- Quality metric: silhouette score = 0.334 (moderate separation)

Summary:

# Pages in cluster	Avg violations per page	Avg critical violations per page	Domains	Risk score	Risk rank
4	225.0	0.0	1	225.0	1
0	147.7	3.2	50	154.0	2

3	52.1	0.4	7	53.0	3
1	33.9	0.6	532	35.1	4
2	3.0	0.0	1	3.0	5

Five clusters emerged. Four clusters represent distinct, recurring accessibility problem families, while one cluster contains low-violation or mixed-pattern outlier pages and is not discussed further.

1. Contrast-Heavy Pages
   - Dominated by color contrast violations
   - Particularly harmful for users with low vision
2. Navigation Confusion Pages
   - Missing landmarks and regions
   - Screen reader users struggle to orient and navigate
3. Structure-Broken Pages
   - Duplicate IDs and broken HTML
   - Assistive technologies fail to interpret content correctly
4. Heading Disorder Pages
   - Incorrect heading order
   - Users cannot skim or understand content structure

Implication: This shows that accessibility problems are systematic, not random. These problem clusters enable teams to identify recurring accessibility failure patterns and apply targeted, repeatable fixes rather than addressing violations individually.

Human Impact: Accessibility failures disproportionately affect real people:

Violation Type	**Who It Harms**	**Why**
Color contrast	Low-vision users	Text becomes unreadable
Missing link names	Screen reader users	Links make no sense
Missing landmarks	Blind users	Page navigation breaks
Duplicate IDs	Assistive tech users	Page structure collapses

These issues create invisible but significant barriers to access.

Recommendations: Top fixes with highest impact

1. **Fix color contrast: Violations covered: 747, ~ 21.2% of all violations**
   - Enforce WCAG contrast ratios through automated design
   - Integrate automated contrast testing into design and CI pipelines
2. **Add semantic landmarks: Violations covered: 869, ~24.7% of all violations**
   - Use <nav>, <main>, <footer> consistently
   - Help improve screen reader navigation
3. **Ensure meaningful link text: Violations covered: 423, ~12.0% of all violations**
   - Avoid vague links like “click here”

- Clearly describe the user action and destination
- Include alt text for images used as links

4. **Validate HTML structure: Violations covered: 763, ~21.7% of all violations**
    - Ensure unique IDs
    - Maintain proper heading hierarchy

These fixes are low-cost, high-impact, and prevent accessibility issues before deployment. Addressing just these four design practices could prevent nearly 80% of all accessibility violations (2,802 out of 3,524) in our dataset.

Conclusion

Accessibility violations are not rare and random. They follow clear patterns, affect specific groups, and can be measured and fixed. By combining data-driven analysis, severity based ranking and human centered interpretation, this project demonstrates how organizations can move toward more inclusive digital systems. Accessibility is not a niche compliance issue, it is a measurable design responsibility, and this analysis shows exactly where to begin.

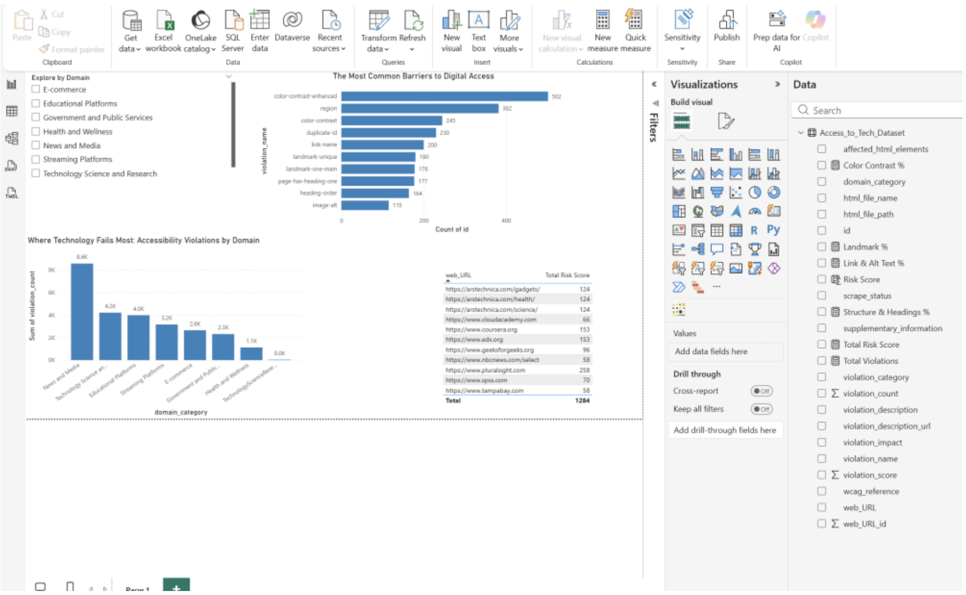
Reproducibility:

All analysis was performed using Python (pandas, visualization libraries) and validated using Power BI. The approach is transparent, reproducible, and designed for reuse.

Code repository: https://github.com/SunayanaHaz/Data4lyf.git

Googlecolab:https://colab.research.google.com/drive/1iclBmpolzQGUzKSsfUM2cDxYLt6KgrAG?usp=sharing

Team Name: Data4Lyf
Team Members: **Sunayana Hazarika, JP Lai, Gaukhar Zharkeyeva, Jacob Edelson**
DubsTech Datathon 2026 Track: Technology



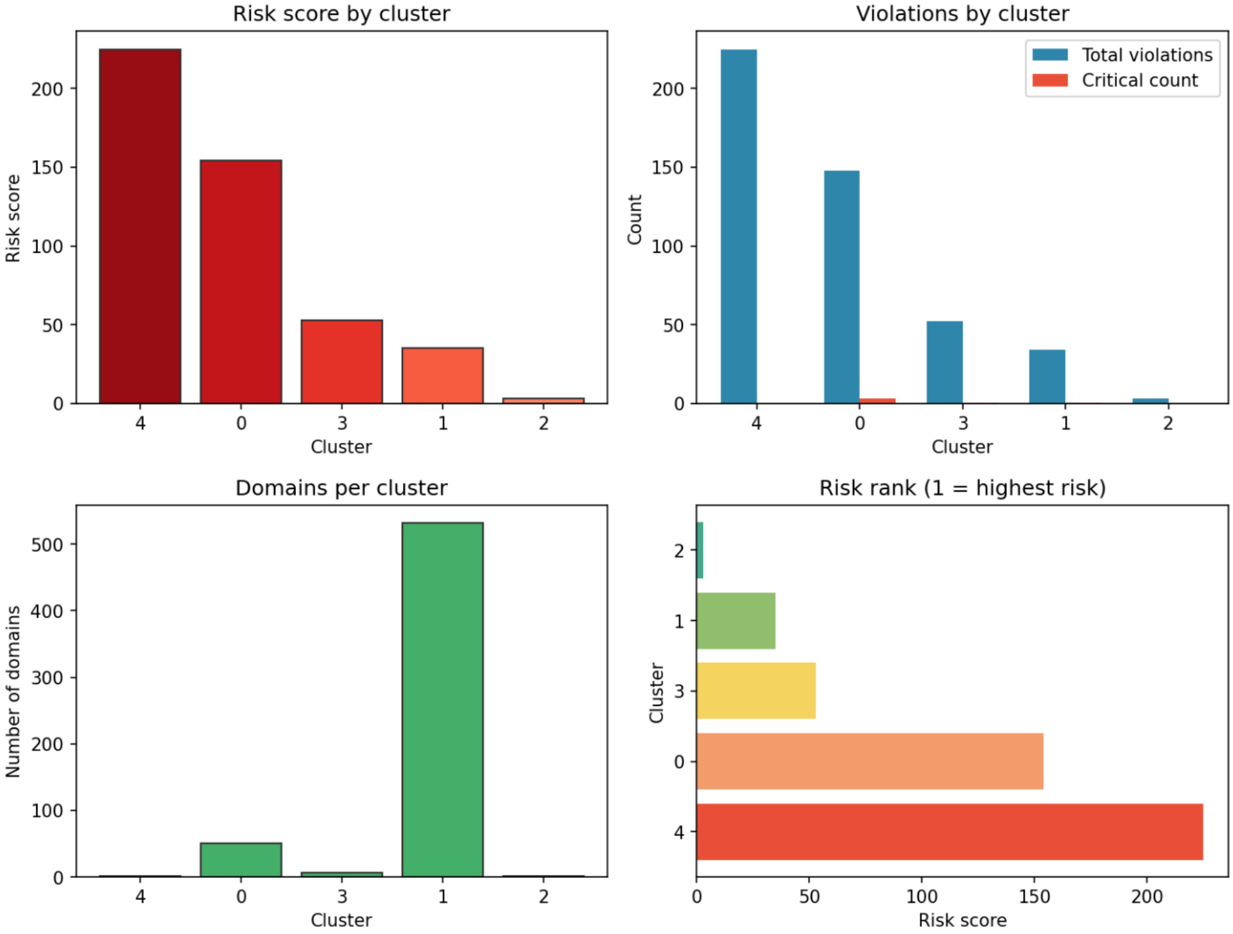
Explore by Domain

- □ E-commerce
- □ Educational Platforms
- □ Government and Public Services
- □ Health and Wellness
- □ News and Media
- □ Streaming Platforms
- □ Technology Science and Research

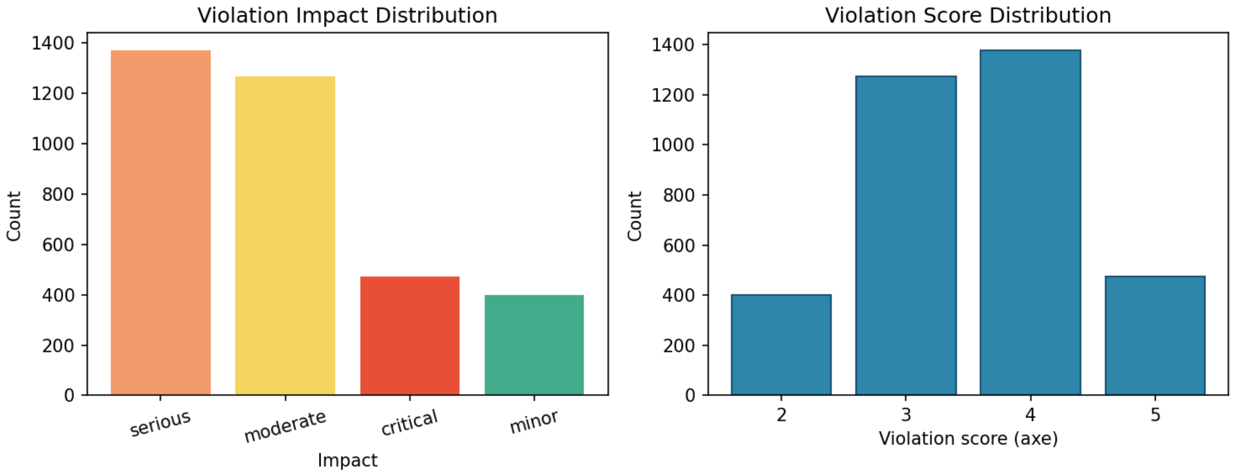
web_URL	Total Risk Score
https://arstechnica.com/gadgets/	124
https://arstechnica.com/health/	124
https://arstechnica.com/science/	124
https://www.cloudacademy.com	66
https://www.coursera.org	153
https://www.edx.org	153
https://www.geeksforgeeks.org	96
https://www.nbcnews.com/select	58
https://www.pluralsight.com	258
https://www.spss.com	70
https://www.tampabay.com	58
Total	**1284**

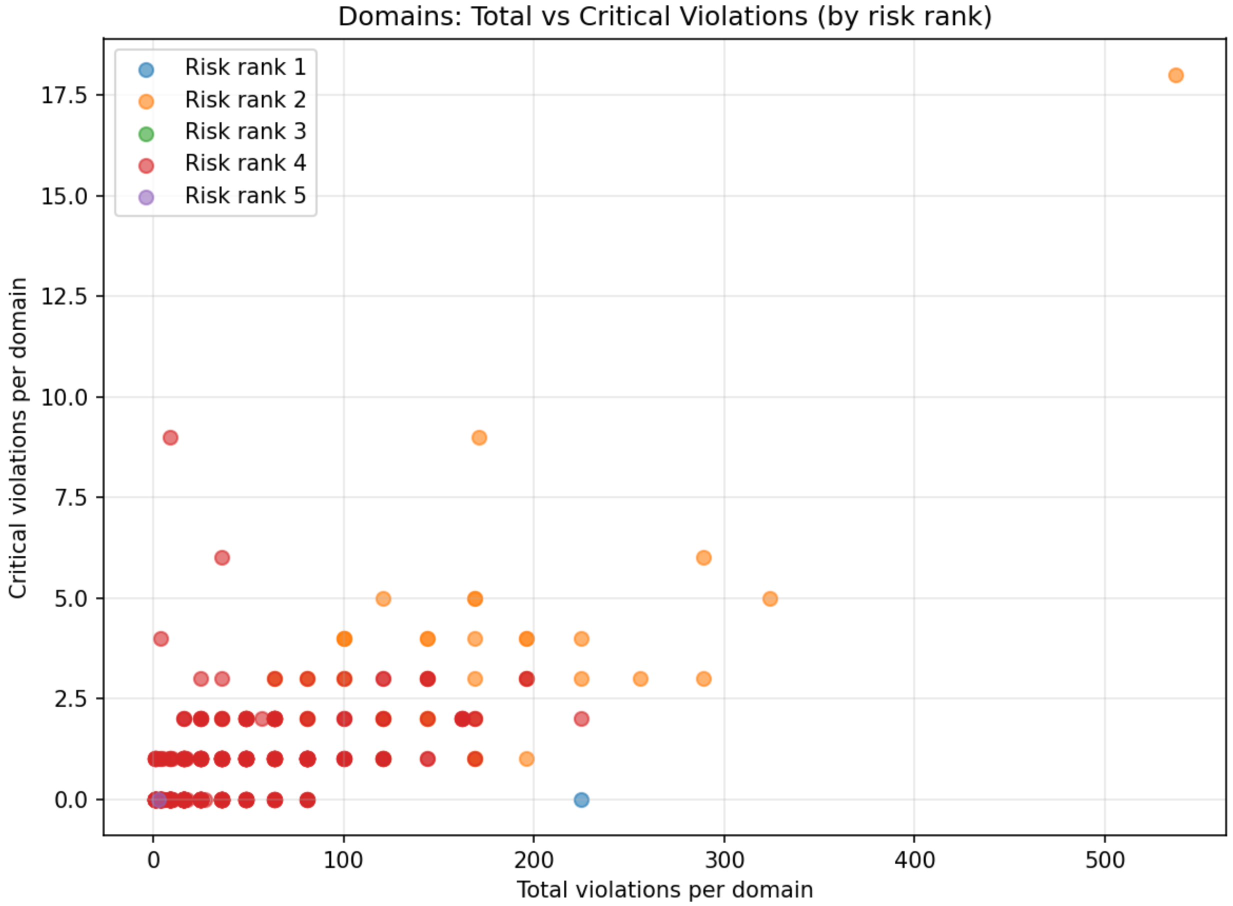
Power BI Dashboard

Cluster Risk Summary

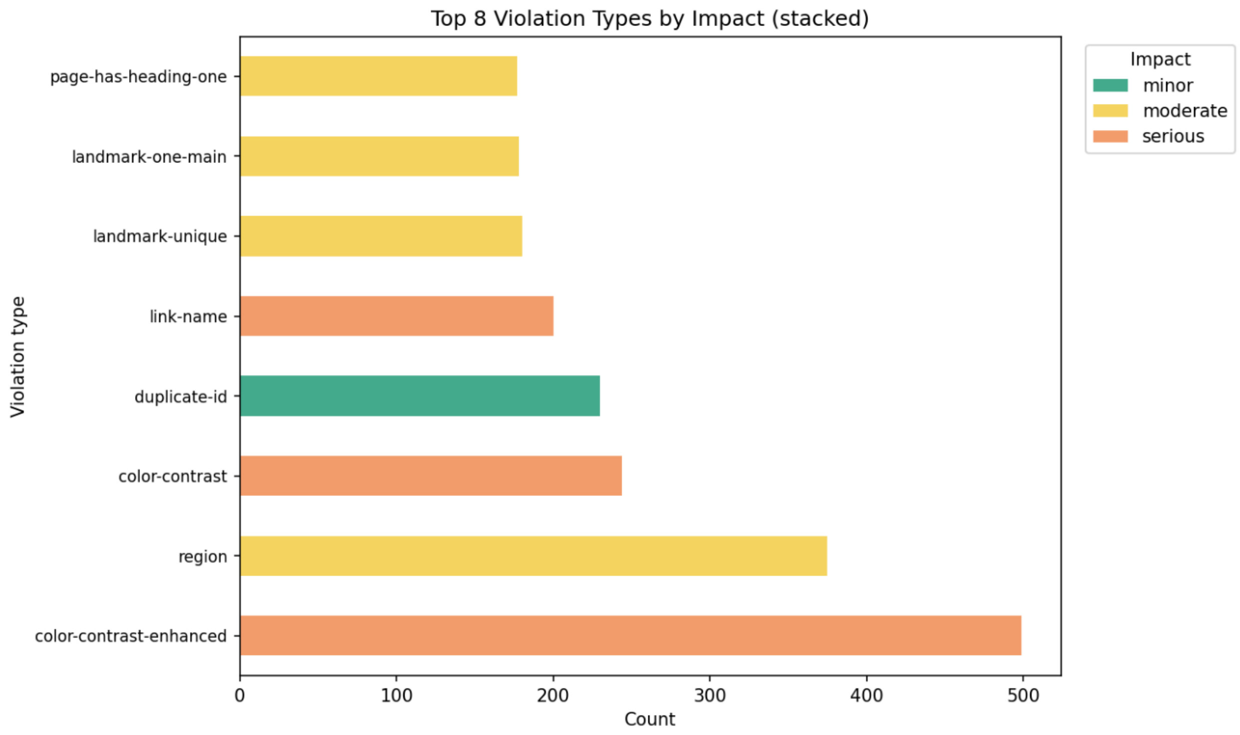


Team Name: Data4Lyf
Team Members: **Sunayana Hazarika, JP Lai, Gaukhar Zharkeyeva, Jacob Edelson**
DubsTech Datathon 2026 Track: Technology

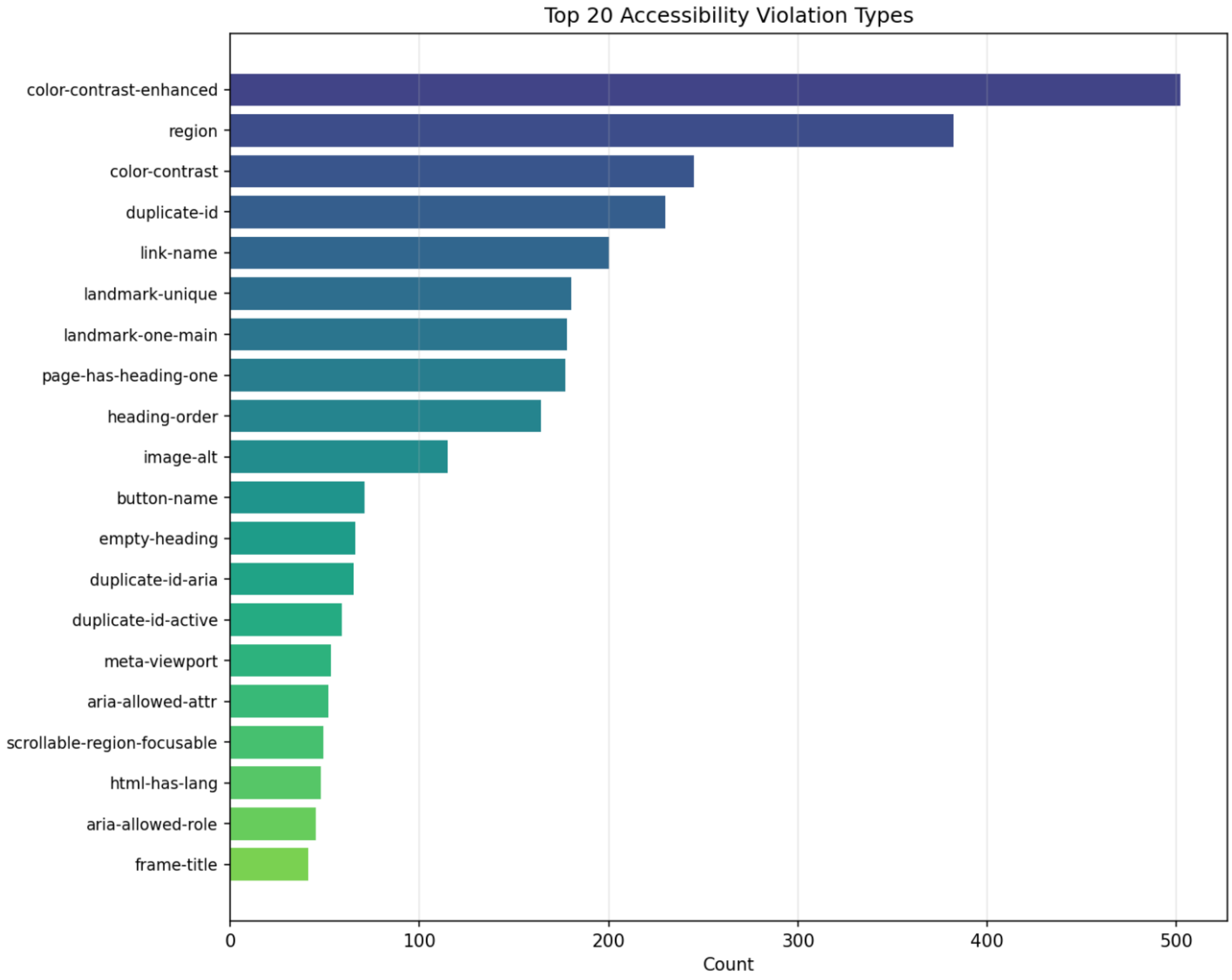




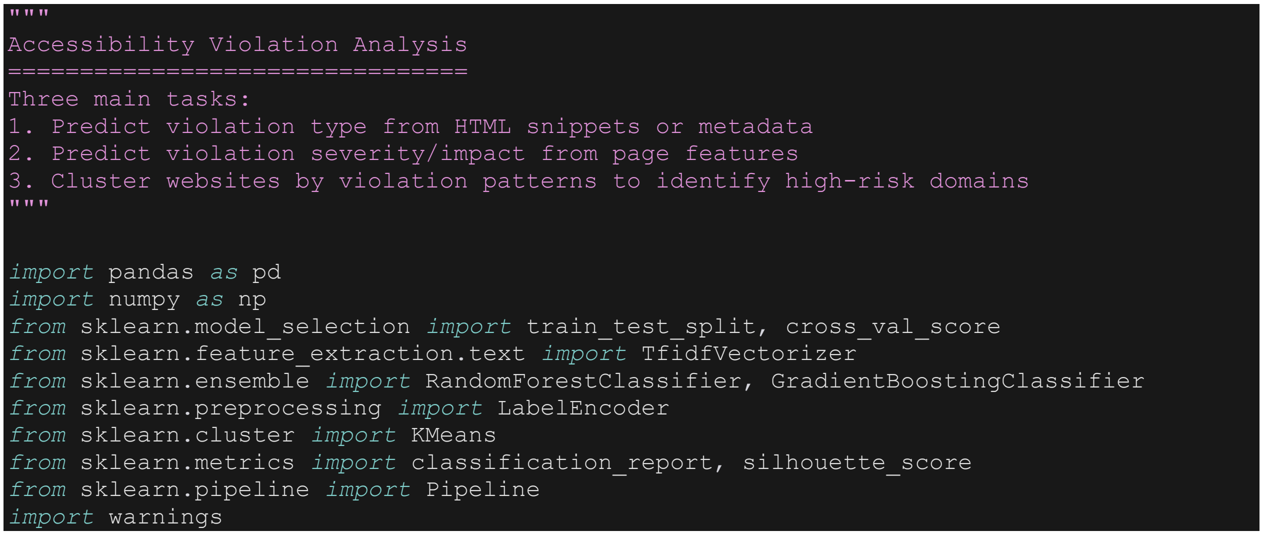
Team Name: Data4Lyf
Team Members: **Sunayana Hazarika, JP Lai, Gaukhar Zharkeyeva, Jacob Edelson**
DubsTech Datathon 2026 Track: Technology



Team Name: Data4Lyf
Team Members: **Sunayana Hazarika, JP Lai, Gaukhar Zharkeyeva, Jacob Edelson**
DubsTech Datathon 2026 Track: Technology

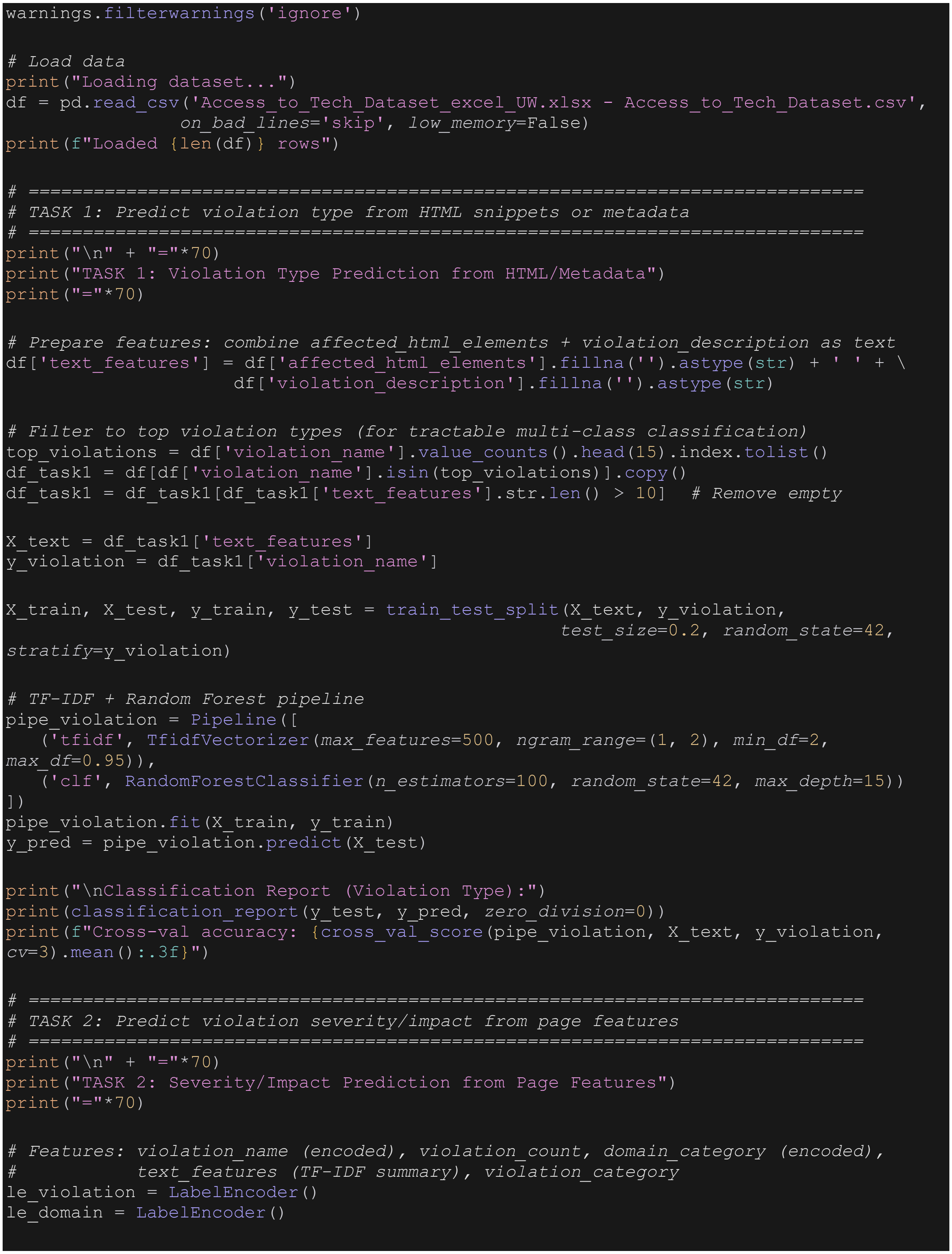


Appendix: Code



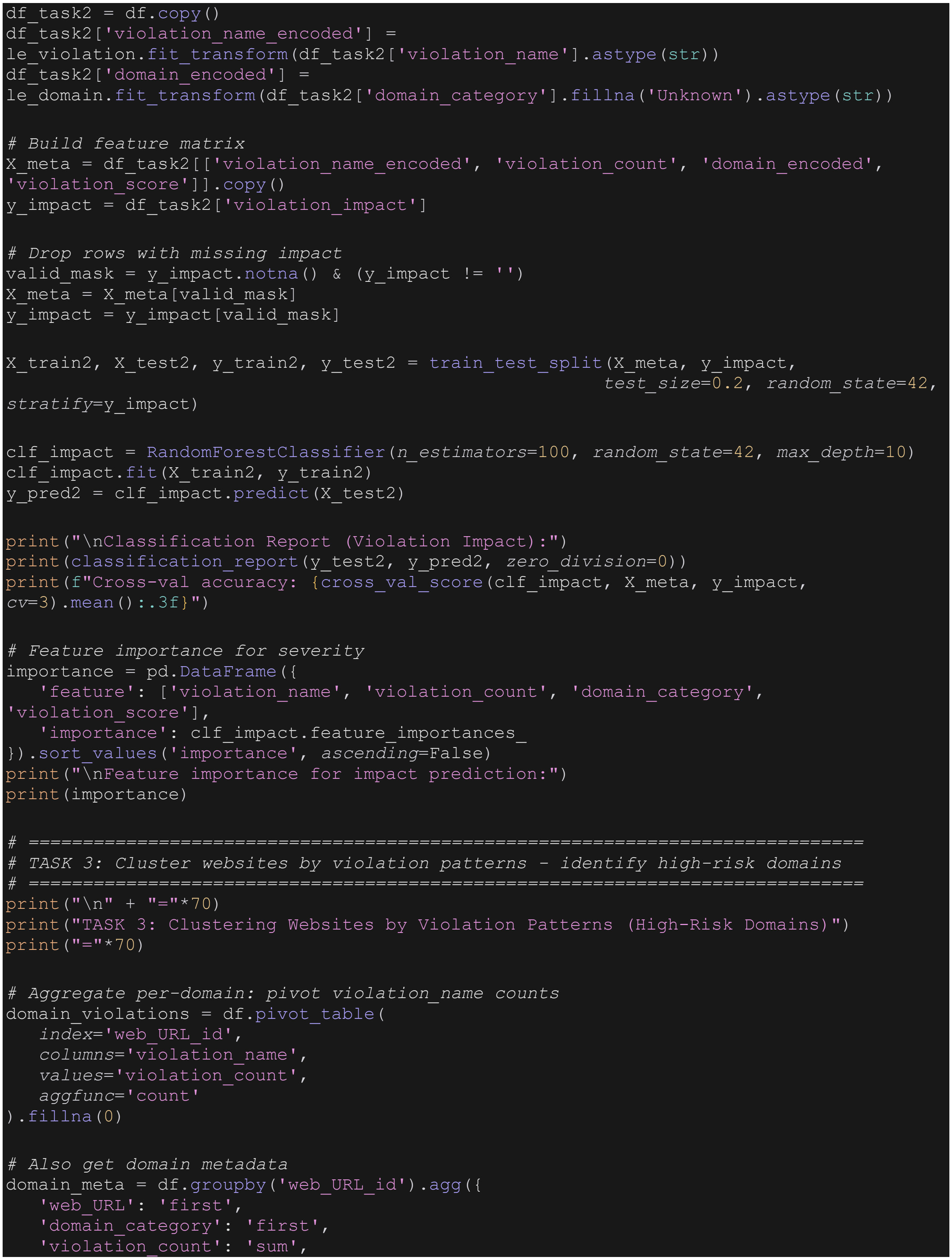
```
"""
Accessibility Violation Analysis
================================
Three main tasks:
1. Predict violation type from HTML snippets or metadata
2. Predict violation severity/impact from page features
3. Cluster websites by violation patterns to identify high-risk domains
"""


import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import KMeans
from sklearn.metrics import classification_report, silhouette_score
from sklearn.pipeline import Pipeline
import warnings
```

Team Name: Data4Lyf
Team Members: **Sunayana Hazarika, JP Lai, Gaukhar Zharkeyeva, Jacob Edelson**
DubsTech Datathon 2026 Track: Technology

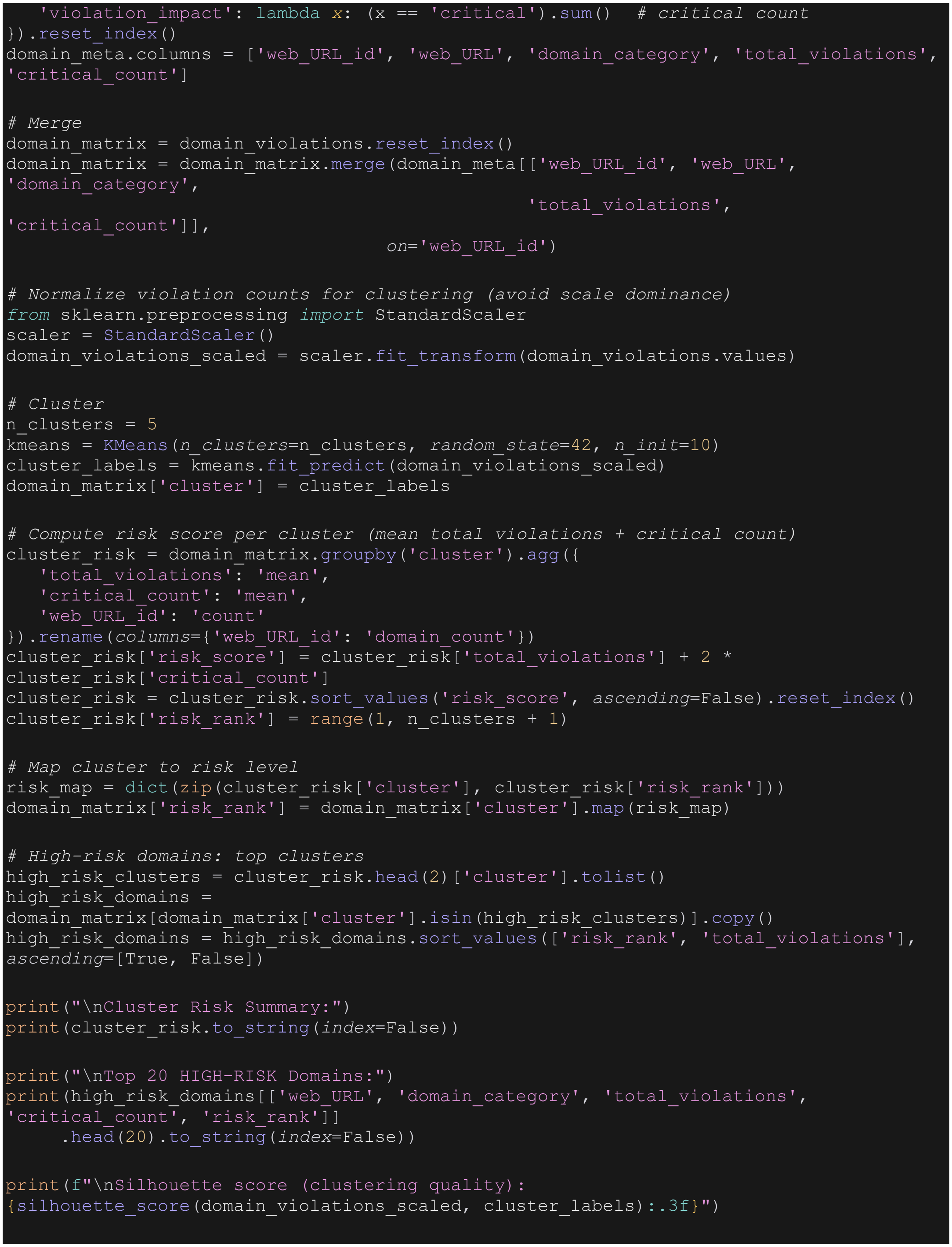


```
warnings.filterwarnings('ignore')

# Load data
print("Loading dataset...")
df = pd.read_csv('Access_to_Tech_Dataset_excel_UW.xlsx - Access_to_Tech_Dataset.csv',
                 on_bad_lines='skip', low_memory=False)
print(f"Loaded {len(df)} rows")

# ==============================================================================
# TASK 1: Predict violation type from HTML snippets or metadata
# ==============================================================================
print("\n" + "="*70)
print("TASK 1: Violation Type Prediction from HTML/Metadata")
print("="*70)

# Prepare features: combine affected_html_elements + violation_description as text
df['text_features'] = df['affected_html_elements'].fillna('').astype(str) + ' ' + \
                      df['violation_description'].fillna('').astype(str)

# Filter to top violation types (for tractable multi-class classification)
top_violations = df['violation_name'].value_counts().head(15).index.tolist()
df_task1 = df[df['violation_name'].isin(top_violations)].copy()
df_task1 = df_task1[df_task1['text_features'].str.len() > 10]  # Remove empty

X_text = df_task1['text_features']
y_violation = df_task1['violation_name']

X_train, X_test, y_train, y_test = train_test_split(X_text, y_violation,
                                                    test_size=0.2, random_state=42,
stratify=y_violation)

# TF-IDF + Random Forest pipeline
pipe_violation = Pipeline([
   ('tfidf', TfidfVectorizer(max_features=500, ngram_range=(1, 2), min_df=2,
max_df=0.95)),
   ('clf', RandomForestClassifier(n_estimators=100, random_state=42, max_depth=15))
])
pipe_violation.fit(X_train, y_train)
y_pred = pipe_violation.predict(X_test)

print("\nClassification Report (Violation Type):")
print(classification_report(y_test, y_pred, zero_division=0))
print(f"Cross-val accuracy: {cross_val_score(pipe_violation, X_text, y_violation,
cv=3).mean():.3f}")

# ==============================================================================
# TASK 2: Predict violation severity/impact from page features
# ==============================================================================
print("\n" + "="*70)
print("TASK 2: Severity/Impact Prediction from Page Features")
print("="*70)

# Features: violation_name (encoded), violation_count, domain_category (encoded),
#           text_features (TF-IDF summary), violation_category
le_violation = LabelEncoder()
le_domain = LabelEncoder()
```

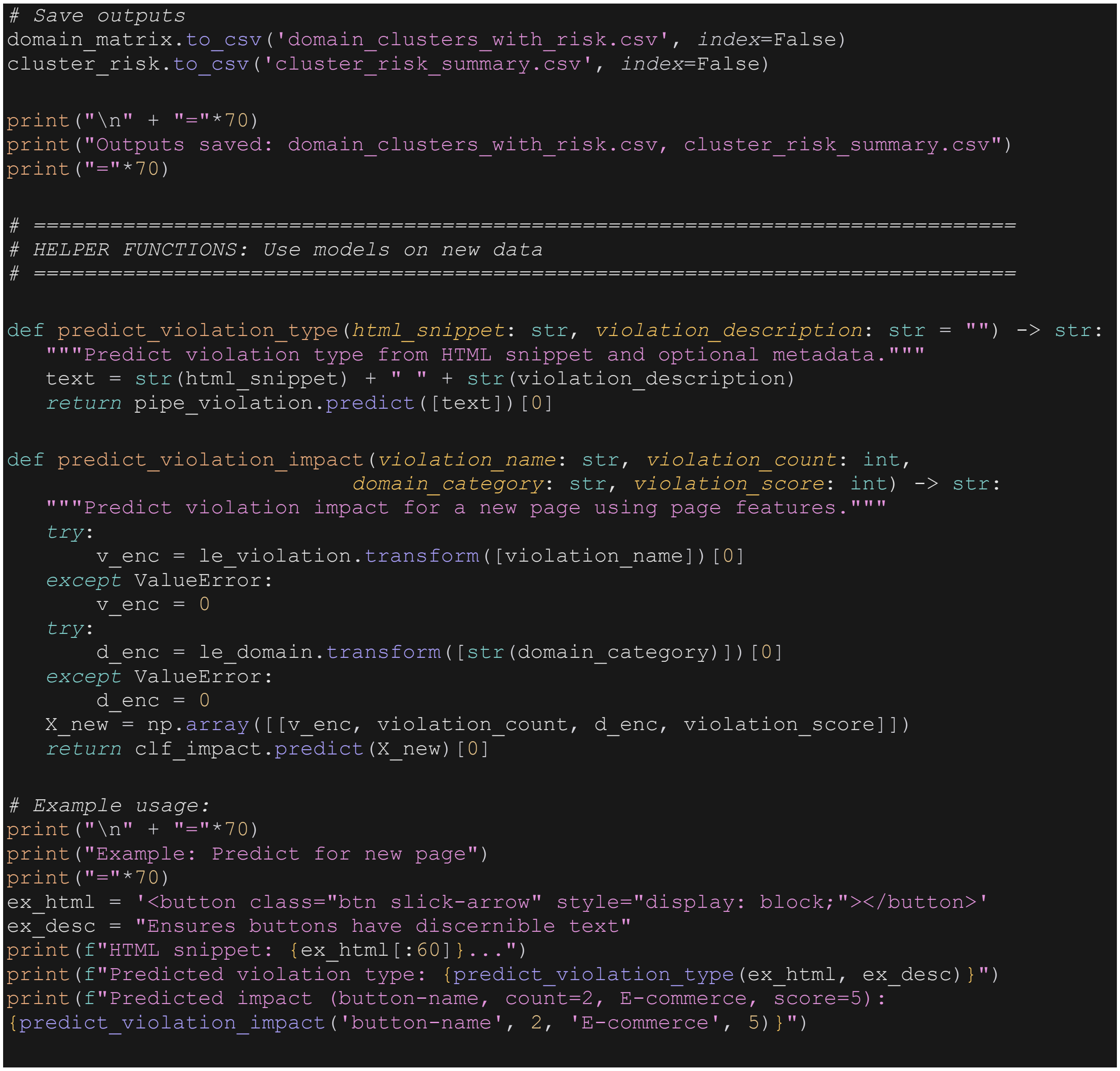
Team Name: Data4Lyf
Team Members: **Sunayana Hazarika, JP Lai, Gaukhar Zharkeyeva, Jacob Edelson**
DubsTech Datathon 2026 Track: Technology



```
df_task2 = df.copy()
df_task2['violation_name_encoded'] =
le_violation.fit_transform(df_task2['violation_name'].astype(str))
df_task2['domain_encoded'] =
le_domain.fit_transform(df_task2['domain_category'].fillna('Unknown').astype(str))

# Build feature matrix
X_meta = df_task2[['violation_name_encoded', 'violation_count', 'domain_encoded',
'violation_score']].copy()
y_impact = df_task2['violation_impact']

# Drop rows with missing impact
valid_mask = y_impact.notna() & (y_impact != '')
X_meta = X_meta[valid_mask]
y_impact = y_impact[valid_mask]

X_train2, X_test2, y_train2, y_test2 = train_test_split(X_meta, y_impact,
                                                       test_size=0.2, random_state=42,
stratify=y_impact)

clf_impact = RandomForestClassifier(n_estimators=100, random_state=42, max_depth=10)
clf_impact.fit(X_train2, y_train2)
y_pred2 = clf_impact.predict(X_test2)

print("\nClassification Report (Violation Impact):")
print(classification_report(y_test2, y_pred2, zero_division=0))
print(f"Cross-val accuracy: {cross_val_score(clf_impact, X_meta, y_impact,
cv=3).mean():.3f}")

# Feature importance for severity
importance = pd.DataFrame({
   'feature': ['violation_name', 'violation_count', 'domain_category',
'violation_score'],
   'importance': clf_impact.feature_importances_
}).sort_values('importance', ascending=False)
print("\nFeature importance for impact prediction:")
print(importance)

# ======================================================================
# TASK 3: Cluster websites by violation patterns - identify high-risk domains
# ======================================================================
print("\n" + "="*70)
print("TASK 3: Clustering Websites by Violation Patterns (High-Risk Domains)")
print("="*70)

# Aggregate per-domain: pivot violation_name counts
domain_violations = df.pivot_table(
   index='web_URL_id',
   columns='violation_name',
   values='violation_count',
   aggfunc='count'
).fillna(0)

# Also get domain metadata
domain_meta = df.groupby('web_URL_id').agg({
   'web_URL': 'first',
   'domain_category': 'first',
   'violation_count': 'sum',
```



```
   'violation_impact': lambda x: (x == 'critical').sum()  # critical count
}).reset_index()
domain_meta.columns = ['web_URL_id', 'web_URL', 'domain_category', 'total_violations',
'critical_count']

# Merge
domain_matrix = domain_violations.reset_index()
domain_matrix = domain_matrix.merge(domain_meta[['web_URL_id', 'web_URL',
'domain_category',
                                                'total_violations',
'critical_count']],
                                   on='web_URL_id')

# Normalize violation counts for clustering (avoid scale dominance)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
domain_violations_scaled = scaler.fit_transform(domain_violations.values)

# Cluster
n_clusters = 5
kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
cluster_labels = kmeans.fit_predict(domain_violations_scaled)
domain_matrix['cluster'] = cluster_labels

# Compute risk score per cluster (mean total violations + critical count)
cluster_risk = domain_matrix.groupby('cluster').agg({
   'total_violations': 'mean',
   'critical_count': 'mean',
   'web_URL_id': 'count'
}).rename(columns={'web_URL_id': 'domain_count'})
cluster_risk['risk_score'] = cluster_risk['total_violations'] + 2 *
cluster_risk['critical_count']
cluster_risk = cluster_risk.sort_values('risk_score', ascending=False).reset_index()
cluster_risk['risk_rank'] = range(1, n_clusters + 1)

# Map cluster to risk level
risk_map = dict(zip(cluster_risk['cluster'], cluster_risk['risk_rank']))
domain_matrix['risk_rank'] = domain_matrix['cluster'].map(risk_map)

# High-risk domains: top clusters
high_risk_clusters = cluster_risk.head(2)['cluster'].tolist()
high_risk_domains =
domain_matrix[domain_matrix['cluster'].isin(high_risk_clusters)].copy()
high_risk_domains = high_risk_domains.sort_values(['risk_rank', 'total_violations'],
ascending=[True, False])

print("\nCluster Risk Summary:")
print(cluster_risk.to_string(index=False))

print("\nTop 20 HIGH-RISK Domains:")
print(high_risk_domains[['web_URL', 'domain_category', 'total_violations',
'critical_count', 'risk_rank']]
     .head(20).to_string(index=False))

print(f"\nSilhouette score (clustering quality):
{silhouette_score(domain_violations_scaled, cluster_labels):.3f}")
```



```
# Save outputs
domain_matrix.to_csv('domain_clusters_with_risk.csv', index=False)
cluster_risk.to_csv('cluster_risk_summary.csv', index=False)

print("\n" + "="*70)
print("Outputs saved: domain_clusters_with_risk.csv, cluster_risk_summary.csv")
print("="*70)

# ==============================================================================
# HELPER FUNCTIONS: Use models on new data
# ==============================================================================

def predict_violation_type(html_snippet: str, violation_description: str = "") -> str:
   """Predict violation type from HTML snippet and optional metadata."""
   text = str(html_snippet) + " " + str(violation_description)
   return pipe_violation.predict([text])[0]

def predict_violation_impact(violation_name: str, violation_count: int,
                             domain_category: str, violation_score: int) -> str:
   """Predict violation impact for a new page using page features."""
   try:
       v_enc = le_violation.transform([violation_name])[0]
   except ValueError:
       v_enc = 0
   try:
       d_enc = le_domain.transform([str(domain_category)])[0]
   except ValueError:
       d_enc = 0
   X_new = np.array([[v_enc, violation_count, d_enc, violation_score]])
   return clf_impact.predict(X_new)[0]

# Example usage:
print("\n" + "="*70)
print("Example: Predict for new page")
print("="*70)
ex_html = '<button class="btn slick-arrow" style="display: block;"></button>'
ex_desc = "Ensures buttons have discernible text"
print(f"HTML snippet: {ex_html[:60]}...")
print(f"Predicted violation type: {predict_violation_type(ex_html, ex_desc)}")
print(f"Predicted impact (button-name, count=2, E-commerce, score=5):
{predict_violation_impact('button-name', 2, 'E-commerce', 5)}")
```