Final Project Aviation Incident Classification and Topic Modeling

This projects aims to train a classification model to predict the main issues and categories of ASRS incident reports. Topic modeling will be used to uncover themes in the same reports. The dataset is sourced from NASA's Aviation Safety Reporting System which contains free-text incident narratives.

In [1]:
```
# Load Libraries here
import pandas as pd
import re
import matplotlib.pyplot as plt
import nltk
import numpy as np
import pyLDAvis
import pyLDAvis.lda_model
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from collections import Counter
from nltk import FreqDist
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 accuracy_score, classification_report
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, classification_report, ConfusionMatrixDisplay
from sklearn.decomposition import NMF
from sklearn.decomposition import TruncatedSVD
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from gensim import corpora
from gensim.models import CoherenceModel
from gensim.models.ldamulticore import LdaMulticore
from tqdm import tqdm
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\16302/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to C:\Users\16302/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to C:\Users\16302/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package punkt to C:\Users\16302/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Load Dataset

In [2]:
```
# Load the CSV (adjust path if needed)
df = pd.read_csv("data/ASRS_DBOnline.csv")
```

Clean, Tokenize & Normalize

In [3]:
```
# 2.1 Drop any empty narratives
df = df.dropna(subset=['Report 1'])

# 2.2 Lower-case all text
df['clean_text'] = df['Report 1'].str.lower()

# 2.3 Remove everything except letters & spaces
df['clean_text'] = df['clean_text'].str.replace(r'[^a-z\s]', ' ', regex=True)

# 2.4 Tokenize by splitting on whitespace
df['tokens'] = df['clean_text'].str.split()
```

```
# 2.5 Remove stop words and 1-letter tokens
stop = set(stopwords.words('english'))
df['tokens'] = df['tokens'].apply(
    lambda toks: [t for t in toks if t not in stop and len(t) > 1]
)

# 2.6 Lemmatize for normalization
lemmatizer = WordNetLemmatizer()
df['tokens'] = df['tokens'].apply(
    lambda toks: [lemmatizer.lemmatize(t) for t in toks]
)
```

In [4]:
```
df[['Report 1','clean_text','tokens']].head()
```

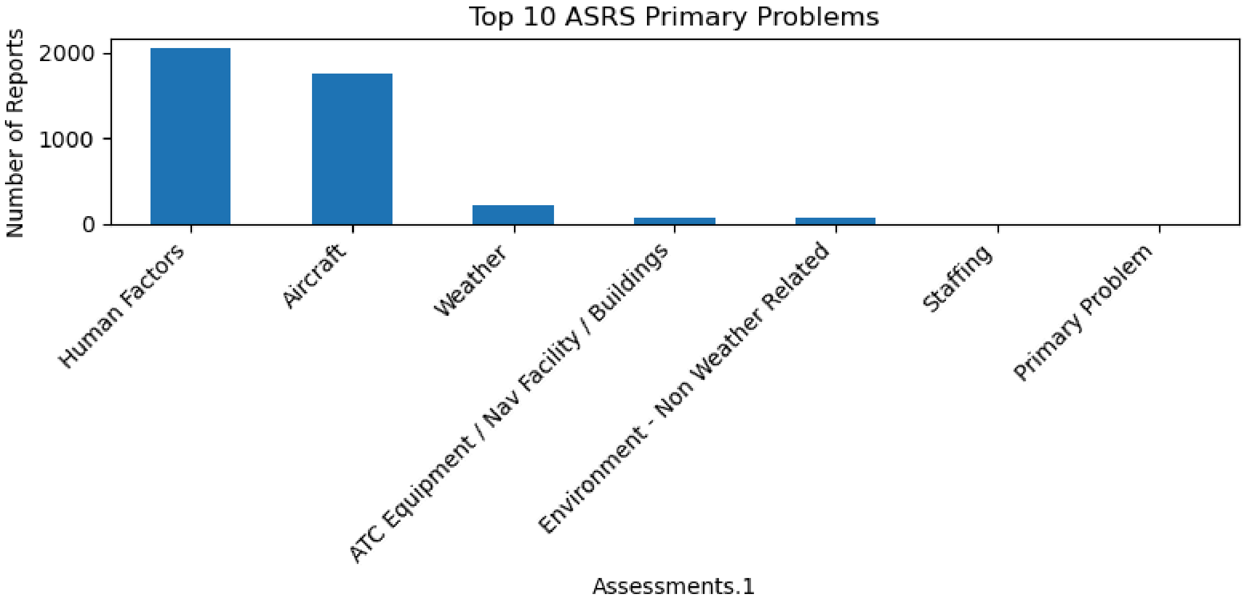
Out[4]:

	Report 1	clean_text	tokens
0	Narrative	narrative	[narrative]
1	Was told to line up and wait runway XXR at int...	was told to line up and wait runway xxr at int...	[told, line, wait, runway, xxr, intersection, ...
2	A large corporate aircraft taxied with in 5-8 ...	a large corporate aircraft taxied with in ...	[large, corporate, aircraft, taxied, inch, sta...
3	GPS Spoofing. Enroute today from ZZZ to SOF; w...	gps spoofing enroute today from zzz to sof w...	[gps, spoofing, enroute, today, zzz, sof, nort...
4	Practicing simulated arcs with instrument stud...	practicing simulated arcs with instrument stud...	[practicing, simulated, arc, instrument, stude...

Exploratory Data Analysis

In [5]:
```
# Count how many reports per problem category
label_counts = df['Assessments.1'].value_counts()

# Display top 10
print(label_counts.head(10))

# Bar plot of top 10
label_counts.head(10).plot(kind='bar', figsize=(8,4))
plt.title("Top 10 ASRS Primary Problems")
plt.ylabel("Number of Reports")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

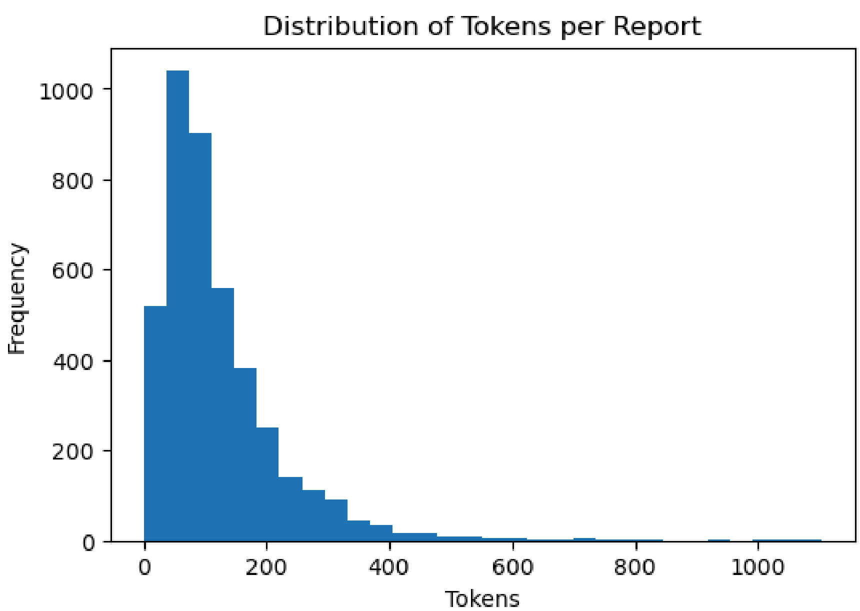
```
Assessments.1
Human Factors                            2055
Aircraft                                 1751
Weather                                   218
ATC Equipment / Nav Facility / Buildings   70
Environment - Non Weather Related          67
Staffing                                    2
Primary Problem                             1
Name: count, dtype: int64
```



Human Factors and Aircraft account for nearly 90% of the reports.

In [6]:
```
# Compute token counts
df['token_count'] = df['tokens'].map(len)

print(df['token_count'].describe())

df['token_count'].plot(kind='hist', bins=30, figsize=(6,4))
plt.title("Distribution of Tokens per Report")
plt.xlabel("Tokens")
plt.show()
```

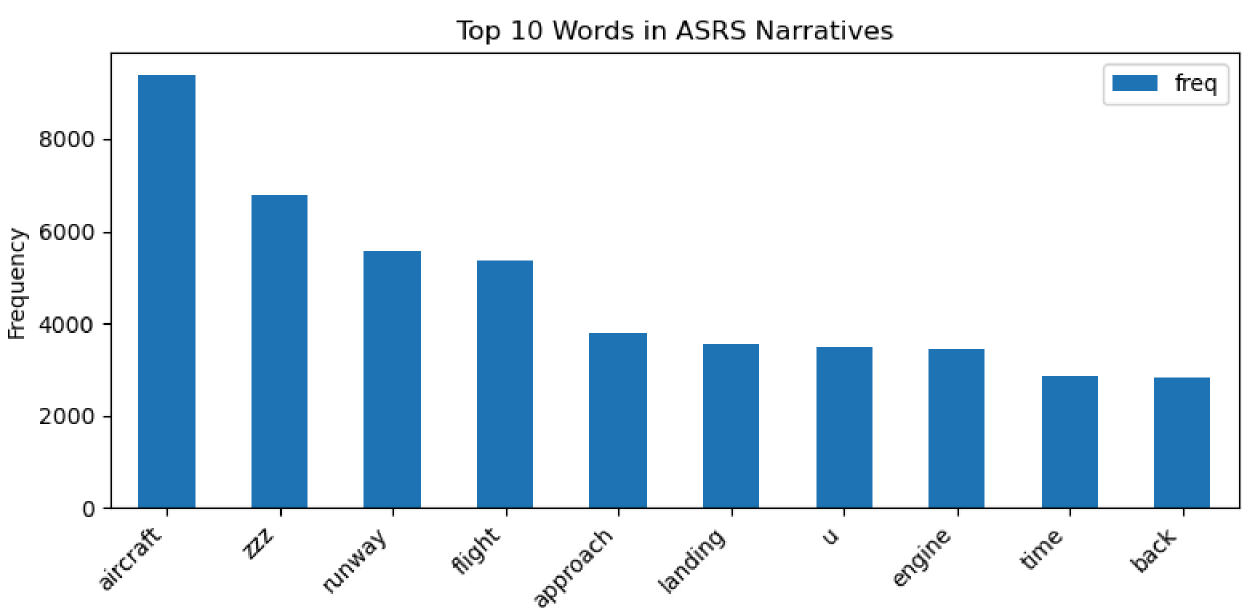
```
count    4164.000000
mean      123.274015
std       104.814701
min         1.000000
25%        57.000000
50%        95.000000
75%       157.000000
max      1102.000000
Name: token_count, dtype: float64
```



Narratives average about 123 Tokens with a majority falling between 60 and 160 words. This makes sense as radio communications should be "short and sweet".

In [7]:
```
# Flatten all tokens into one list and count
all_counts = Counter()
df['tokens'].map(all_counts.update)

# Convert to DataFrame
freq_df = (
    pd.DataFrame.from_dict(all_counts, orient='index', columns=['freq'])
      .sort_values('freq', ascending=False)
)

# Show top 20
print(freq_df.head(20))

# Plot top 10
freq_df.head(10).plot(kind='bar', figsize=(8,4))
plt.title("Top 10 Words in ASRS Narratives")
plt.ylabel("Frequency")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
             freq
aircraft     9395
zzz          6799
runway       5553
flight       5346
approach     3781
landing      3547
u            3482
engine       3459
time         2868
back         2814
pilot        2797
would        2519
left         2443
altitude     2413
right        2323
atc          2257
control      2187
airport      2084
captain      1932
maintenance  1905
```



ZZZ suggests further cleaning. Aircraft, Runway, Flight, Approach reflect reports consisting of routine aircraft operations and runway interactions.

Create Features and Labels -

Use preprocessed tokens to create features and use Assessments.1 as the label; build feature words based on frequency

In [8]:
```
# Build feature words based on frequency
word_cutoff = 5
tokens = [t for tokens in df['tokens'] for t in tokens]
word_dist = FreqDist(tokens)

feature_words = set()
for word, count in word_dist.items():
    if count >= word_cutoff:
        feature_words.add(word)
```

Define feature extraction function

In [9]:
```
def conv_features(text, fw):
    """Convert text to feature dictionary for NLTK Naive Bayes"""
    text_set = set(text.split())
    text_set = text_set.intersection(fw)
    return {word: True for word in text_set}
```

Clean data and build feature sets

In [10]:
```
# Ensure there are no nulls in tokens
df = df.dropna(subset=['tokens'])

# Join token list into a single string
df['joined_tokens'] = df['tokens'].apply(lambda x: ' '.join(x))

# Convert joined tokens into feature sets
featuresets = [
    (conv_features(text, feature_words), label)
    for text, label in zip(df['joined_tokens'], df['Assessments.1'])
]
```

Naive Bayes

Train/Test Split and train Naive Bayes Classifier

In [11]:
```
import random
random.seed(42)
random.shuffle(featuresets)

test_size = 500
test_set = featuresets[:test_size]
train_set = featuresets[test_size:]

classifier = nltk.NaiveBayesClassifier.train(train_set)
```

In [12]:
```
# Accuracy
print("Naive Bayes Accuracy:", nltk.classify.accuracy(classifier, test_set))

# Most informative features
classifier.show_most_informative_features(10)
```

```
Naive Bayes Accuracy: 0.006
Most Informative Features
                creating = True           Staffi : Aircra =    852.2 : 1.0
                division = True           Staffi : Human  =    606.3 : 1.0
              procedural = True           Staffi : Human  =    606.3 : 1.0
                 staffed = True           Staffi : Human  =    606.3 : 1.0
                  uneven = True           Staffi : Human  =    606.3 : 1.0
                shortcut = True           Staffi : Aircra =    511.3 : 1.0
             supervision = True           Staffi : Aircra =    511.3 : 1.0
               narrative = True           Primar : Human  =    389.8 : 1.0
                   angry = True           Staffi : Aircra =    306.8 : 1.0
                handoff = True            Staffi : Aircra =    306.8 : 1.0
```

Based on the accuracy, Naive Bayes fails to generalize and likely predicts only the most common class or guesses randomly.

Predict and Compare with actual labels

In [13]:
```
import textwrap

# Start from index 1 to skip the first "narrative" placeholder
for i in range(1, 6):  # Rows 1 to 5
    report = df.iloc[i]['joined_tokens']
    wrapped_report = textwrap.fill(report, width=100)  # Adjust line width as needed
    predicted = classifier.classify(conv_features(report, feature_words))
    actual = df.iloc[i]['Assessments.1']

    print(f"Report:\n{wrapped_report}")
    print(f"Predicted: {predicted} | Actual: {actual}\n")
```

```
Report:
told line wait runway xxr intersection cleared take passing kt pic pilot flying rejected takeoff due
tow crossing approach end runway xxl sight picture looked like aircraft tow crossing mid runway
rejected takeoff pic pilot flying exited runway taxiway taxied back run checklist zzz tower might
wanted advise u would crossing aircraft tow approach end runway clearing u takeoff tug towing
aircraft might cleared time takeoff unsure due u tower frequency ground frequency
Predicted: Primary Problem | Actual: Human Factors

Report:
large corporate aircraft taxied inch static parked helicopter monitoring ramp situation lack
corporate pilot awareness taxi maneuver tao regional airport systemically unsatisfactory past
previous collision occurring static helicopter transient taxing corporate jet rushed outside became
apparent clearance issue taxing jet static helicopter might compromised unable signal corporate
pilot unwilling stop aircraft appeared clearance jet wing tip helicopter rotor blade might
compromised aircraft taxi speed faster brisk walk fbo ground guide present jet aircraft taxing
normal flow ramp traffic direction suggestion ramp area mismanaged parked aircraft transient taxi
aircraft upon discussion management suggested ramp area front hangar designated long term fixed wing
parking area ground marshaling required transient aircraft congested area information could included
awos broadcast afd multiple signage notams require fbo agree policy procedure additionally fbo
manned night hour mitigation helicopter remain inside hangar night hazard light could installed
around parked helicopter
Predicted: Primary Problem | Actual: Human Factors

Report:
gps spoofing enroute today zzz sof north egypt approaching lakto intersection filed route ipads
showed airplane directly olba beirut international airport event lasted minute verified exact
location airplane course airplane never course ipads affected
Predicted: Primary Problem | Actual: ATC Equipment / Nav Facility / Buildings

Report:
practicing simulated arc instrument student aircraft came directly u altitude descended turned avoid
aircraft followed closely sped depart aircraft continued following entered mode veil aircraft adsd
communicating freq
Predicted: Primary Problem | Actual: Human Factors

Report:
departed zzz ppl training control normal take cruise maneuver simulated loss power procedure
throttle seemed getting slightly harder move regardless friction lock setting flew zzz pattern work
pattern work throttle usability decreased power full power achievable modulation possible climb
black gray oil seen throttle rod returned zzz full power advised tower issue flew high fast approach
full power setting throttle idle landing without power taxied back tie down alternating full power
idle shut without incident damage aircraft injury people
Predicted: Primary Problem | Actual: Aircraft
```

Model overpredicts label "Primary Problem" (label bias) and fails to identify patterns in content-rich narratives as shown by obvious human factors examples.

Logistic Regression

Vectorize text with TF-IDF

In [14]:
```
# Filter out rare labels (keep only labels with >= 5 samples)
label_counts = df['Assessments.1'].value_counts()
valid_labels = label_counts[label_counts >= 5].index
df_filtered = df[df['Assessments.1'].isin(valid_labels)].copy()
```

```
# Prepare features and labels
X = df_filtered['joined_tokens']
y = df_filtered['Assessments.1']

# TF-IDF Vectorization
vectorizer = TfidfVectorizer(max_features=5000)
X_vec = vectorizer.fit_transform(X)

# Train-test split (stratified)
X_train, X_test, y_train, y_test = train_test_split(
    X_vec, y, test_size=0.2, stratify=y, random_state=42
)
```

Train Logistic Regression

In [15]:
```
# Train a Logistic Regression classifier on the TF-IDF features
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)

# Predict and evaluate
y_pred = lr.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Logistic Regression Accuracy: 0.8703481392557023
                                          precision    recall  f1-score   support

ATC Equipment / Nav Facility / Buildings       0.25      0.07      0.11        14
                                Aircraft       0.88      0.90      0.89       351
       Environment - Non Weather Related       0.00      0.00      0.00        13
                           Human Factors       0.87      0.95      0.91       411
                                 Weather       0.95      0.43      0.59        44

                                accuracy                           0.87       833
                               macro avg       0.59      0.47      0.50       833
                            weighted avg       0.85      0.87      0.85       833
```

The LR model performs well on the categories of Aircraft and Human Factors, but does not perform well on under-represented classes like the Environment-Non Weather and ATC Equipment / Nav Facility / Buildings. Weather has high precision and low recall which means the model is selective/cautious when predicting Weather and will predict it only when it is confident.

Determine Most Informative Features

In [16]:
```
# Extract feature names and model coefficients for interpretation
feature_names = vectorizer.get_feature_names_out()
coefs = lr.coef_

# For multiclass — loop through classes
for idx, class_label in enumerate(lr.classes_):
    top_features = np.argsort(coefs[idx])[-10:]  # Top 10
    print(f"\nTop features for class: {class_label}")
    for feat in reversed(top_features):
        print(f"{feature_names[feat]:>15} : {coefs[idx][feat]:.3f}")
```

```
Top features for class: ATC Equipment / Nav Facility / Buildings
            gps : 2.273
      frequency : 1.544
         sector : 1.500
            rnp : 1.096
             il : 1.034
     glideslope : 1.000
          radio : 0.975
     controller : 0.960
            tag : 0.793
       position : 0.738

Top features for class: Aircraft
         engine : 3.342
            zzz : 2.651
    maintenance : 2.306
            qrh : 2.088
        failure : 1.951
           gear : 1.946
       priority : 1.740
       pressure : 1.731
          cabin : 1.717
         normal : 1.695

Top features for class: Environment - Non Weather Related
            gps : 2.314
           bird : 2.154
       spoofing : 1.193
        terrain : 1.175
         strike : 0.931
        spoofed : 0.903
        jamming : 0.784
         object : 0.704
            mud : 0.690
         relief : 0.683

Top features for class: Human Factors
          drone : 2.004
        traffic : 1.625
            bag : 1.487
             dg : 1.473
            set : 1.459
          final : 1.393
      dangerous : 1.378
         loaded : 1.199
        cleared : 1.183
       pushback : 1.147

Top features for class: Weather
     turbulence : 3.955
         severe : 2.719
        weather : 2.717
           wind : 2.192
    encountered : 1.944
      downdraft : 1.854
      condition : 1.831
           gust : 1.574
           wave : 1.322
       airspeed : 1.291
```

Logistic Regression assigns highest weights to terms that are semantically and operationally relevant to each class as well as shows strong learning of aviation context.

Predict and Compare with actual labels

In [17]:

```
import textwrap

# Filter out rows with placeholder text like 'narrative'
filtered_df = df[df['joined_tokens'].str.lower() != 'narrative']

# Sample 5 full rows so we keep both text and label
sample_rows = filtered_df.sample(5, random_state=42)
```

```
for _, row in sample_rows.iterrows():
    text = row['joined_tokens']
    actual = row['Assessments.1']

    # Wrap text
    wrapped_text = textwrap.fill(text, width=100)

    # Predict
    X_sample = vectorizer.transform([text])
    pred = lr.predict(X_sample)[0]

    # Display
    print(f"Report:\n{wrapped_text}")
    print(f"Predicted: {pred} | Actual: {actual}\n")
```

```
Report:
day flight returning ferry flight zzz zzz approximately xa fl requested priority handling zzz center
due cabin pressurization issue made prior reviewing quick reference handbook qrh training received
following request began descent prior notification descent pressurization controlled fl however
decided descend ft remain prevention gap confirming cabin control informed zzz center requested
leave xxxx squawk mode proceeded destination lower altitude ensuring safety using oxygen mask
landing safely zzz reported incident maintenance department address pressurization issue
investigation maintenance action performed aircraft next flight
Predicted: Aircraft | Actual: Aircraft

Report:
initial approach radar vector precautionary engine shutdown due unable control engine thrust lever
Predicted: Aircraft | Actual: Aircraft

Report:
approximately xa aircraft ifr pc requested clearance zzz airport local controller producing new atis
advised aircraft repeat request local control advised aircraft ifr flight plan system aircraft asked
give ifr information could entered na aircraft issued taxi instruction runway xx promptly issued ifr
clearance weather condition low ifr ceiling mist obstructing view vehicle maintenance departure end
runway xx rwy xx vehicle occupied memory aid use local controller mistakenly overlooked memory aid
local controller realized aircraft cleared runway xx vehicle occupied aircraft already midway runway
aircraft reported vehicle runway local control advised would reported
Predicted: Human Factors | Actual: Human Factors

Report:
taildragger upon landing light crosswind right touched right tire veered runway left braked stop
facing roughly runway xx degree turn left runway heading grass next runway able taxi power right
turn back runway taxi hangar without incident overnight right tire lost air determined tire bead
injury aircraft pattern uncontrolled zzz
Predicted: Human Factors | Actual: Human Factors

Report:
undesired aircraft state due crm breakdown poor automation management following event remember know
perfect flight executed routine test flight evaluate vhf static airbus neo approach briefed rnav xx
zzz mention made would likely visual approach backed rnav descent checklist executed per sop upon
return zzz area controller queried whether flight would like rnav xx visual pf elected execute
visual rnav backup flight cleared direct zzz roughly lined rnav xx seemed easy time pm even
mentioned going direct zzzzz clearance approx mile field descent thousand traffic reported seen tcas
clock mile climbing heading roughly direction much slower pm communicated around time load gas need
knock approach try due traffic could basically fly next hour flight elected level atc aware traffic
sight overtaken flight able descend cloud sct bkn layer approx mile field pm reported field sight
flight cleared visual approach runway xx appr selected pm properly sequenced still direct zzz fmgc
first big error pf called flap selectedas flight approached mile additional traffic reported co
altitude clock mile closing pm diverted attention traffic pf continued share focus traffic approach
pf dealt confusion approach sequenced correctly around time pm became aware automation looking
correct said something effect fly pm intent communicate pf fly approach manually pm said something
like click fly pitch thrust manually pf disconnected autopilot took throttle clb detent disconnect
autothrottles pf asked pm set altitude minimum tdze pm set something lower set based box sequenced
based sop matter point pm still diverting attention ensuring flight need go around traffic pm recall
whether airspeed still managed mile field pm gained sight traffic flight switched tower pm looked
towards runway perceived high rate descent crosschecking instrument ft min rod noted pm said
something like ft min rate descent flight approached msl pm communicated flight low pm could tell pf
confused automation asked want turn flight director somewhere around time event sequence may
slightly jumbled based time compression stress aircraft got slow knot definitely yellow band
maneuvering speed speed speed speed aural alert sounded pm noted pf increased throttle angle failed
notice associated increase thrust autothrottles still engaged also point sequence pf asked flap pm
selected flap pf selected higher thrust lever setting aircraft sped climbed pm assessed safe
approach possible stated think go around pf called go around climbed landing pattern set altitude
box airspeed selected point tower queried flight would like go back center flight stayed tower
executed left closed pattern landing runway xx
Predicted: Human Factors | Actual: Human Factors
```

Logistic Regression predicted all 5 reports correctly with narratives of different complexities, demonstrating ability to generalize well. With the TF-IDF, this helped the model understand context on top of word frequency.

Support Vector Machine

In [18]:
```
# Train the SVM model
svm = LinearSVC()
svm.fit(X_train, y_train)
```

```
# Predict and evaluate
y_pred_svm = svm.predict(X_test)

print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_svm))
```

```
SVM Accuracy: 0.8787515006002401

Classification Report:
                                          precision    recall  f1-score   support

ATC Equipment / Nav Facility / Buildings       0.29      0.14      0.19        14
                                Aircraft       0.89      0.91      0.90       351
       Environment - Non Weather Related       0.50      0.15      0.24        13
                           Human Factors       0.88      0.94      0.91       411
                                 Weather       0.96      0.55      0.70        44

                                accuracy                           0.88       833
                               macro avg       0.70      0.54      0.59       833
                            weighted avg       0.87      0.88      0.87       833
```

The SVM model performs similarly well also on the categories of Aircraft and Human Factors and not well to under-represented classes like the Environment-Non Weather and ATC Equipment / Nav Facility / Buildings. Weather has high precision and low recall which means the model is selective/cautious when predicting Weather and will predict it only when it is confident. SVM is a stronger classfier if maximizing accuracy and handling minority classes.

Determine Most informative features - SVM

In [19]:
```
# Extract feature names and model coefficients for interpretation
feature_names = vectorizer.get_feature_names_out()
coefs = svm.coef_

# For multiclass — loop through classes
for idx, class_label in enumerate(svm.classes_):
    top10 = np.argsort(coefs[idx])[-10:]
    print(f"\nTop features for class: {class_label}")
    for feat_index in reversed(top10):
        print(f"{feature_names[feat_index]:>15} : {coefs[idx][feat_index]:.3f}")
```

```
Top features for class: ATC Equipment / Nav Facility / Buildings
         sector : 1.029
            bgr : 1.024
            rnp : 1.004
     glideslope : 0.996
            cpr : 0.951
            tag : 0.916
            sbn : 0.909
        goggles : 0.862
             il : 0.860
       position : 0.830

Top features for class: Aircraft
        failure : 2.053
         engine : 1.990
            qrh : 1.697
            rpm : 1.606
       pressure : 1.555
            zzz : 1.508
    maintenance : 1.505
        problem : 1.463
          issue : 1.416
           fail : 1.367

Top features for class: Environment - Non Weather Related
           bird : 1.487
        spoofed : 1.317
       spoofing : 1.116
            mud : 1.086
          cairo : 1.055
            gps : 0.882
            orl : 0.813
     electronic : 0.808
        terrain : 0.799
      eastbound : 0.798

Top features for class: Human Factors
          drone : 1.597
     distracted : 1.556
            set : 1.538
           solo : 1.452
            bag : 1.420
           task : 1.405
           wake : 1.388
         rather : 1.253
          drove : 1.240
         loaded : 1.234

Top features for class: Weather
         severe : 2.352
     turbulence : 2.268
        weather : 1.904
      downdraft : 1.841
    encountered : 1.559
           gust : 1.479
           snow : 1.424
          slide : 1.374
           wave : 1.257
      condition : 1.176
```

The SVM model effectively captures key aviation terms with the ability to separate different classes based on the content rich narratives, identify meaningful patterns in addition to term frequency, and has high accuracy, demonstating its robustness in performing the task.

Predict and Compare with actual labels

In [20]:

```
import textwrap

# Sample and skip the first row
for i, row in df.iloc[1:].sample(5, random_state=42).iterrows():
    full_text = row['joined_tokens']
    wrapped_text = "\n".join(textwrap.wrap(full_text, width=100))  # wrap for readability
    X_sample = vectorizer.transform([full_text])
    pred = lr.predict(X_sample)[0]
```

```
    actual = row['Assessments.1']

    print("Report:")
    print(wrapped_text, end='')  # no newline added at the end
    print(f"\nPredicted: {pred} | Actual: {actual}\n")
```

```
Report:
day flight returning ferry flight zzz zzz approximately xa fl requested priority handling zzz center
due cabin pressurization issue made prior reviewing quick reference handbook qrh training received
following request began descent prior notification descent pressurization controlled fl however
decided descend ft remain prevention gap confirming cabin control informed zzz center requested
leave xxxx squawk mode proceeded destination lower altitude ensuring safety using oxygen mask
landing safely zzz reported incident maintenance department address pressurization issue
investigation maintenance action performed aircraft next flight
Predicted: Aircraft | Actual: Aircraft

Report:
initial approach radar vector precautionary engine shutdown due unable control engine thrust lever
Predicted: Aircraft | Actual: Aircraft

Report:
approximately xa aircraft ifr pc requested clearance zzz airport local controller producing new atis
advised aircraft repeat request local control advised aircraft ifr flight plan system aircraft asked
give ifr information could entered na aircraft issued taxi instruction runway xx promptly issued ifr
clearance weather condition low ifr ceiling mist obstructing view vehicle maintenance departure end
runway xx rwy xx vehicle occupied memory aid use local controller mistakenly overlooked memory aid
local controller realized aircraft cleared runway xx vehicle occupied aircraft already midway runway
aircraft reported vehicle runway local control advised would reported
Predicted: Human Factors | Actual: Human Factors

Report:
taildragger upon landing light crosswind right touched right tire veered runway left braked stop
facing roughly runway xx degree turn left runway heading grass next runway able taxi power right
turn back runway taxi hangar without incident overnight right tire lost air determined tire bead
injury aircraft pattern uncontrolled zzz
Predicted: Human Factors | Actual: Human Factors

Report:
undesired aircraft state due crm breakdown poor automation management following event remember know
perfect flight executed routine test flight evaluate vhf static airbus neo approach briefed rnav xx
zzz mention made would likely visual approach backed rnav descent checklist executed per sop upon
return zzz area controller queried whether flight would like rnav xx visual pf elected execute
visual rnav backup flight cleared direct zzz roughly lined rnav xx seemed easy time pm even
mentioned going direct zzzzz clearance approx mile field descent thousand traffic reported seen tcas
clock mile climbing heading roughly direction much slower pm communicated around time load gas need
knock approach try due traffic could basically fly next hour flight elected level atc aware traffic
sight overtaken flight able descend cloud sct bkn layer approx mile field pm reported field sight
flight cleared visual approach runway xx appr selected pm properly sequenced still direct zzz fmgc
first big error pf called flap selectedas flight approached mile additional traffic reported co
altitude clock mile closing pm diverted attention traffic pf continued share focus traffic approach
pf dealt confusion approach sequenced correctly around time pm became aware automation looking
correct said something effect fly pm intent communicate pf fly approach manually pm said something
like click fly pitch thrust manually pf disconnected autopilot took throttle clb detent disconnect
autothrottles pf asked pm set altitude minimum tdze pm set something lower set based box sequenced
based sop matter point pm still diverting attention ensuring flight need go around traffic pm recall
whether airspeed still managed mile field pm gained sight traffic flight switched tower pm looked
towards runway perceived high rate descent crosschecking instrument ft min rod noted pm said
something like ft min rate descent flight approached msl pm communicated flight low pm could tell pf
confused automation asked want turn flight director somewhere around time event sequence may
slightly jumbled based time compression stress aircraft got slow knot definitely yellow band
maneuvering speed speed speed speed aural alert sounded pm noted pf increased throttle angle failed
notice associated increase thrust autothrottles still engaged also point sequence pf asked flap pm
selected flap pf selected higher thrust lever setting aircraft sped climbed pm assessed safe
approach possible stated think go around pf called go around climbed landing pattern set altitude
box airspeed selected point tower queried flight would like go back center flight stayed tower
executed left closed pattern landing runway xx
Predicted: Human Factors | Actual: Human Factors
```

SVM predicted all 5 report narratives correctly effectively identifying Aircraft-related issues and Human Factors. SVM is able to leverage aviation key terms effectively and able to identify key patterns in the narratives, demonstrating robust performance classifying aviation safety reports.

Topic Modeling

In [21]:
```
#load CSV
df = pd.read_csv("data/ASRS_DBOnline.csv")

#drop missing reports
df = df.dropna(subset=['Report 1'])

#preprocessing setup
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

#preprocess and build joined_tokens
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-z\s]', ' ', text)
    tokens = text.split()
    tokens = [t for t in tokens if t not in stop_words and len(t) > 1]
    tokens = [lemmatizer.lemmatize(t) for t in tokens]
    return ' '.join(tokens)

df['joined_tokens'] = df['Report 1'].apply(clean_text)
```

In [22]:
```
vectorizer = TfidfVectorizer(max_features=5000)
X_tfidf = vectorizer.fit_transform(df['joined_tokens'])
```

Non-negative Matrix Funcation Model

In [23]:
```
#number of topics
num_topics = 10

#fit the NMF model
nmf_model = NMF(n_components=num_topics, random_state=42)
W = nmf_model.fit_transform(X_tfidf)
H = nmf_model.components_
```

In [24]:
```
def display_topics(model, feature_names, top_words=10):
    for topic_idx, topic in enumerate(model.components_):
        print(f"Topic {topic_idx:5d}")
        top_indices = topic.argsort()[::-1][:top_words]
        for i in top_indices:
            print(f"{feature_names[i]} ({topic[i]:.2f})")
        print()


display_topics(nmf_model, vectorizer.get_feature_names_out())
```

```
Topic     0
zzz (1.03)
captain (0.85)
flight (0.77)
maintenance (0.72)
qrh (0.63)
cabin (0.63)
checklist (0.62)
passenger (0.59)
gate (0.56)
dispatch (0.55)

Topic     1
aircraft (1.14)
runway (0.96)
traffic (0.66)
pattern (0.59)
downwind (0.47)
call (0.39)
student (0.39)
xx (0.34)
left (0.33)
radio (0.33)

Topic     2
approach (1.61)
visual (0.58)
altitude (0.52)
terrain (0.45)
cleared (0.40)
final (0.38)
ft (0.35)
tower (0.34)
zzzzz (0.31)
low (0.31)

Topic     3
brake (1.73)
parking (0.75)
tug (0.55)
set (0.55)
push (0.50)
crew (0.41)
aircraft (0.40)
ramp (0.38)
taxiway (0.37)
ground (0.35)

Topic     4
gear (2.14)
landing (0.66)
nose (0.42)
runway (0.36)
main (0.36)
light (0.32)
flap (0.25)
locked (0.24)
extension (0.22)
green (0.22)

Topic     5
engine (2.35)
power (0.64)
oil (0.57)
zzz (0.33)
runway (0.33)
rpm (0.31)
takeoff (0.28)
checklist (0.27)
normal (0.24)
landing (0.24)

Topic     6
dg (1.91)
```

```
dangerous (0.83)
good (0.70)
summary (0.57)
final (0.55)
received (0.50)
acars (0.38)
message (0.36)
sent (0.36)
code (0.35)

Topic     7
fuel (1.76)
tank (0.94)
zzz (0.57)
pump (0.29)
leak (0.25)
left (0.24)
wing (0.21)
center (0.21)
flight (0.20)
plane (0.18)

Topic     8
altitude (0.74)
turbulence (0.71)
drone (0.58)
ft (0.54)
foot (0.47)
aircraft (0.42)
fl (0.39)
atc (0.39)
autopilot (0.38)
climb (0.37)

Topic     9
gps (1.81)
jamming (0.51)
interference (0.38)
spoofing (0.33)
fir (0.30)
navigation (0.29)
position (0.23)
rnp (0.22)
system (0.20)
anp (0.17)
```

Each topic clusters words commonly found together in similar types of incident reports. The weights show how strongly each word contributes to its topic. This model helps surface hidden themes like "landing gear problems" or "GPS interference" across the dataset.

Latent Semantic Analysis

In [25]:
```
#number of topics
num_topics = 10

#fit LSA model
lsa_model = TruncatedSVD(n_components=num_topics, random_state=42)
lsa_model.fit(X_tfidf)

#topic-word matrix
H_lsa = lsa_model.components_
```

In [26]:
```
def display_topics(model, feature_names, top_words=10):
    for topic_idx, topic in enumerate(model.components_):
        print(f"Topic {topic_idx:02d}")
        top_indices = topic.argsort()[::-1][:top_words]
        for i in top_indices:
            print(f"{feature_names[i]} ({topic[i]:.2f})")
        print()


display_topics(lsa_model, vectorizer.get_feature_names_out())
```

```
Topic 00
aircraft (0.26)
runway (0.21)
zzz (0.21)
engine (0.16)
approach (0.15)
flight (0.14)
landing (0.13)
pilot (0.11)
left (0.10)
altitude (0.10)

Topic 01
engine (0.30)
zzz (0.16)
checklist (0.14)
maintenance (0.13)
captain (0.12)
qrh (0.11)
fuel (0.11)
gate (0.10)
crew (0.10)
dispatch (0.09)

Topic 02
approach (0.35)
altitude (0.26)
ft (0.16)
atc (0.14)
terrain (0.12)
visual (0.12)
descent (0.11)
zzzzz (0.10)
autopilot (0.09)
alert (0.09)

Topic 03
engine (0.41)
runway (0.16)
power (0.15)
fuel (0.14)
zzz (0.13)
oil (0.11)
landing (0.10)
student (0.08)
rpm (0.08)
tank (0.07)

Topic 04
gear (0.43)
brake (0.23)
approach (0.21)
runway (0.17)
landing (0.14)
nose (0.11)
taxiway (0.11)
flap (0.10)
speed (0.09)
set (0.08)

Topic 05
gear (0.44)
landing (0.17)
dg (0.14)
zzz (0.12)
runway (0.12)
door (0.10)
pattern (0.10)
message (0.09)
qrh (0.09)
dispatch (0.08)

Topic 06
dg (0.43)
```

```
approach (0.26)
engine (0.23)
final (0.23)
runway (0.22)
dangerous (0.17)
good (0.14)
summary (0.13)
received (0.11)
power (0.11)

Topic 07
gear (0.46)
fuel (0.19)
brake (0.18)
traffic (0.14)
pattern (0.12)
engine (0.12)
tank (0.11)
downwind (0.11)
left (0.09)
tug (0.09)

Topic 08
gps (0.38)
drone (0.17)
turbulence (0.15)
fuel (0.12)
power (0.12)
flight (0.11)
plane (0.11)
jamming (0.11)
airplane (0.10)
student (0.10)

Topic 09
gps (0.65)
jamming (0.18)
aircraft (0.16)
interference (0.14)
engine (0.12)
spoofing (0.12)
fir (0.11)
navigation (0.10)
message (0.10)
position (0.09)
```

LSA uncovered distinct themes in aviation reports, such as engine problems, approach procedures, GPS interference, and landing gear issues. Some topics overlap (e.g. gear/brake in Topics 4, 5, and 7), showing shared vocabulary across scenarios. The model effectively distinguishes between technical, procedural, and environmental concerns.

Latent Dirichlet Allocation

In [27]:

```
#CountVectorizer
count_vectorizer = CountVectorizer(max_features=5000)
count_text_vectors = count_vectorizer.fit_transform(df['joined_tokens'])

#fit LDA model using count-based vectors
lda_model = LatentDirichletAllocation(
    n_components=5,
    random_state=42,
    learning_method='batch'
)
lda_model.fit(count_text_vectors)
```

Out[27]:

```
LatentDirichletAllocation

LatentDirichletAllocation(n_components=5, random_state=42)
```

In [28]:

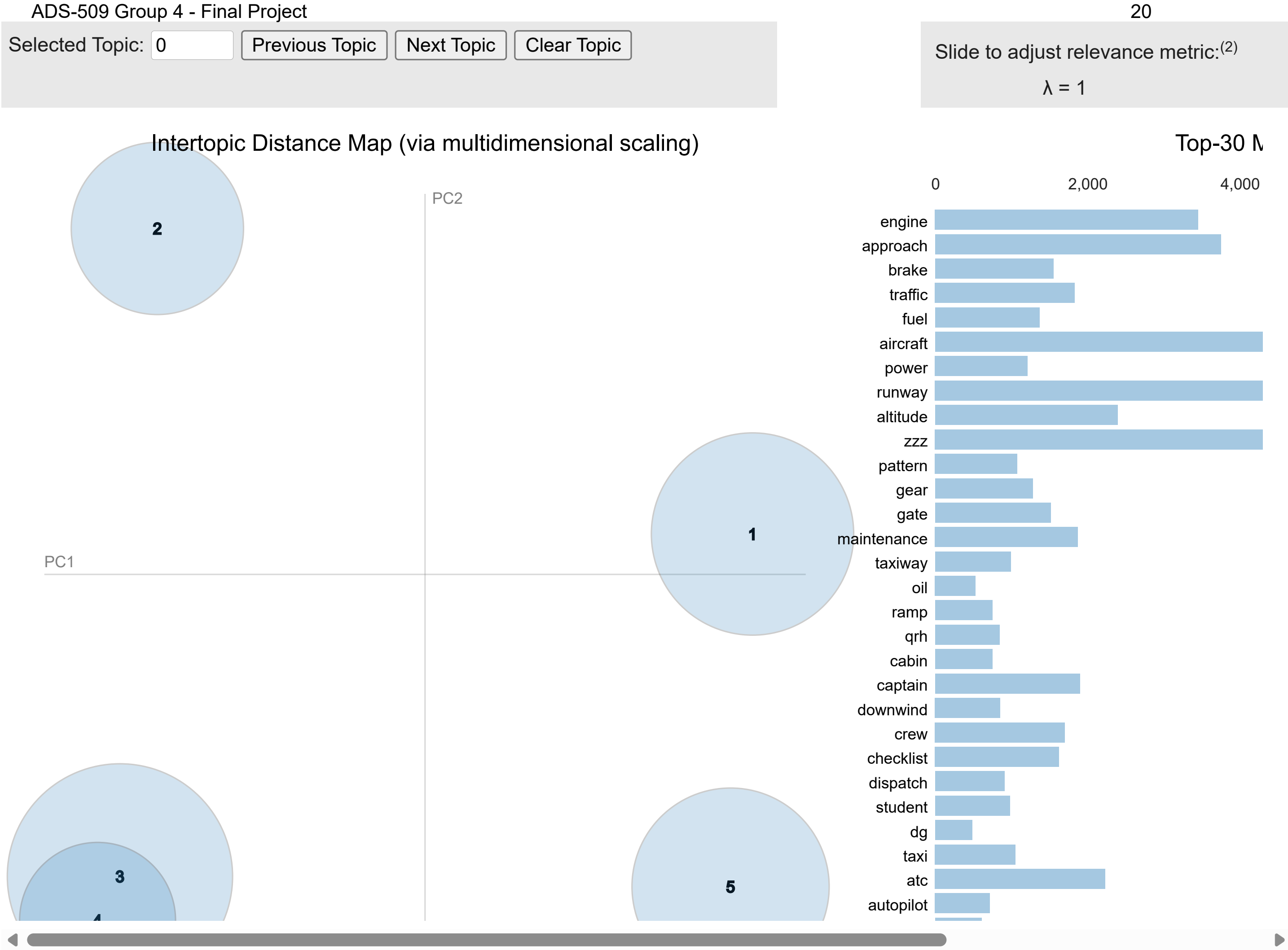
```
#display_topics on fitted model
display_topics(lda_model, count_vectorizer.get_feature_names_out())
```

```
Topic 00
aircraft (4676.38)
runway (2715.04)
traffic (1736.48)
pattern (1046.14)
zzz (1018.94)
tower (985.56)
pilot (944.54)
left (903.63)
turn (876.40)
call (874.83)

Topic 01
aircraft (1784.49)
brake (1487.88)
gear (929.33)
runway (891.52)
taxiway (767.13)
taxi (702.35)
ramp (684.78)
gate (657.02)
back (637.54)
ground (628.80)

Topic 02
zzz (2632.36)
flight (2342.64)
maintenance (1497.28)
aircraft (1397.90)
captain (1308.00)
landing (1295.28)
checklist (1115.56)
control (1012.03)
passenger (1011.59)
would (974.30)

Topic 03
engine (3144.42)
zzz (1690.25)
fuel (1089.76)
power (1003.40)
runway (929.92)
landing (814.43)
flight (707.49)
aircraft (600.06)
back (541.26)
oil (529.74)

Topic 04
approach (2873.09)
altitude (1693.94)
atc (1099.81)
zzz (1071.76)
aircraft (937.16)
ft (924.71)
flight (892.51)
time (702.57)
pilot (686.12)
visual (682.21)
```

LDA surfaced the most statistically dominant themes, such as aircraft control, fuel or engine failures, and approach altitude. The model repeats strong aviation-specific terms (e.g., "aircraft," "runway," "engine"), suggesting it's especially sensitive to frequent domain terms. It offers clear distinctions between flight stage issues, like takeoff (Topic 3), landing (Topic 1), or mid-flight maintenance (Topic 2).

In [29]:
```
#visualize with pyLDAvis
pyLDAvis.enable_notebook()
lda_display = pyLDAvis.lda_model.prepare(
    lda_model,
    count_text_vectors,
    count_vectorizer,
    sort_topics=False
)
lda_display
```

Out[29]:



In [30]:

```
!pip install gensim
```

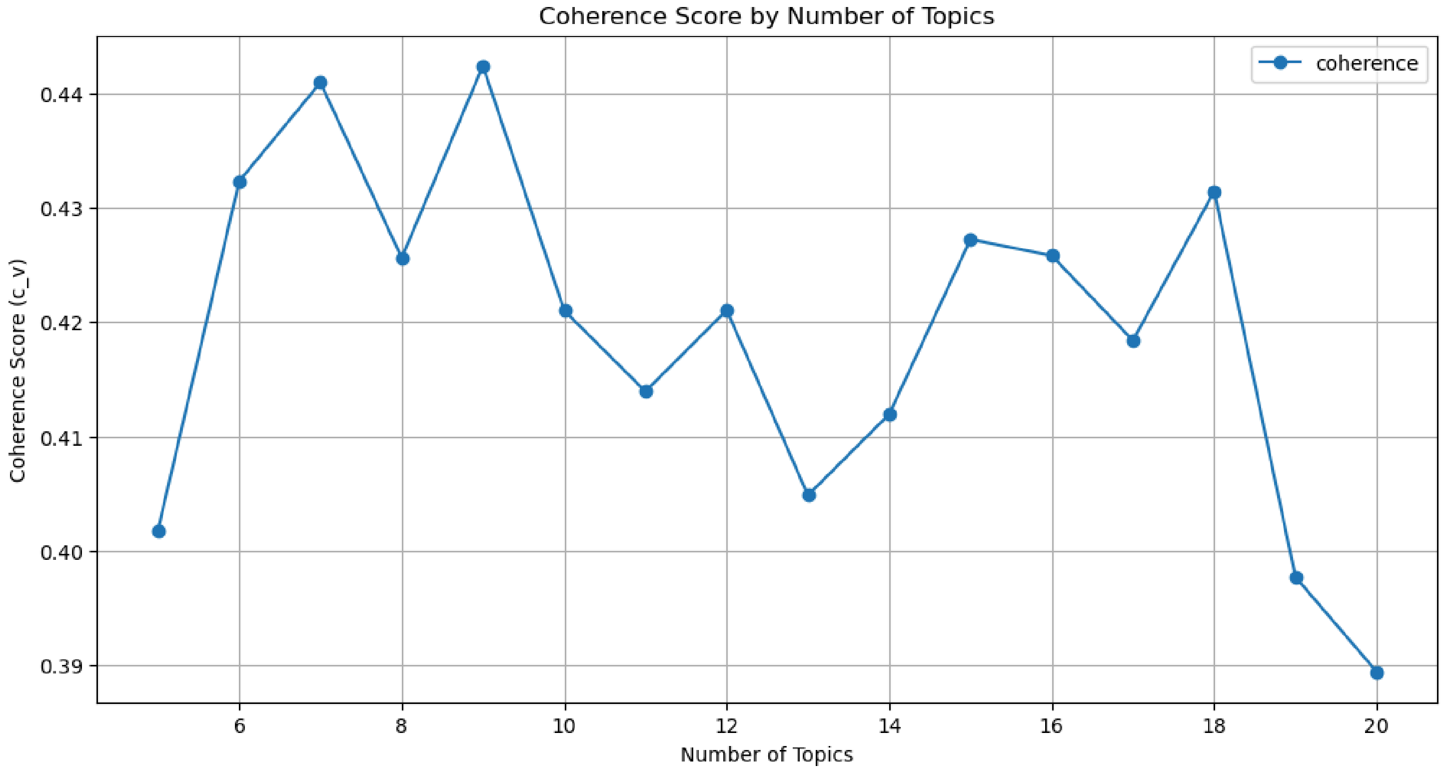
```
Requirement already satisfied: gensim in c:\users\16302\miniconda3\lib\site-packages (4.3.3)
Requirement already satisfied: numpy<2.0,>=1.18.5 in c:\users\16302\miniconda3\lib\site-packages (from gensim) (1.26.4)
Requirement already satisfied: scipy<1.14.0,>=1.7.0 in c:\users\16302\miniconda3\lib\site-packages (from gensim) (1.12.0)
Requirement already satisfied: smart-open>=1.8.1 in c:\users\16302\miniconda3\lib\site-packages (from gensim) (7.1.0)
Requirement already satisfied: wrapt in c:\users\16302\miniconda3\lib\site-packages (from smart-open>=1.8.1->gensim) (1.17.2)
```

In [31]:

```
#rebuild tokens column from joined_tokens
df['tokens'] = df['joined_tokens'].str.split()

#prepare Gensim inputs
gensim_texts = df['tokens'].tolist()
dict_gensim = corpora.Dictionary(gensim_texts)
bow_gensim = [dict_gensim.doc2bow(text) for text in gensim_texts]

#loop over topic numbers and evaluate coherence
lda_para_model_n = []

for n in tqdm(range(5, 21)):
    lda_model = LdaMulticore(
        corpus=bow_gensim,
        id2word=dict_gensim,
        chunksize=2000,
        eta='auto',
        iterations=400,
        num_topics=n,
        passes=20,
        eval_every=None,
        random_state=42
    )

    lda_coherence = CoherenceModel(
        model=lda_model,
        texts=gensim_texts,
        dictionary=dict_gensim,
```

```
        coherence='c_v'
    )

    lda_para_model_n.append((n, lda_model, lda_coherence.get_coherence()))

# Plot coherence scores
pd.DataFrame(lda_para_model_n, columns=["n", "model", "coherence"]) \
  .set_index("n")[["coherence"]] \
  .plot(figsize=(12, 6), marker='o', title='Coherence Score by Number of Topics')

plt.ylabel("Coherence Score (c_v)")
plt.xlabel("Number of Topics")
plt.grid(True)
plt.show()
```

```
100%|████████████████████████████████████████| 16/16 [37:44<00:00, 141.50s/it]
```



Observations:

Best Coherence Score at 9 Topics

The highest coherence score (~0.442) is achieved when the number of topics is 9, suggesting this is the most semantically meaningful and interpretable topic breakdown for your dataset.

Strong Scores at 7, 10, and 18

Topics around 7, 10, and 18 also yield high coherence scores (>0.43), meaning these could also be viable alternatives depending on the granularity you're aiming for.

Sharp Drop After 18 Topics

After 18 topics, coherence drops steeply, indicating the model likely starts overfitting—creating topics that are too narrow or redundant.

Low Scores Below 6 and Above 19

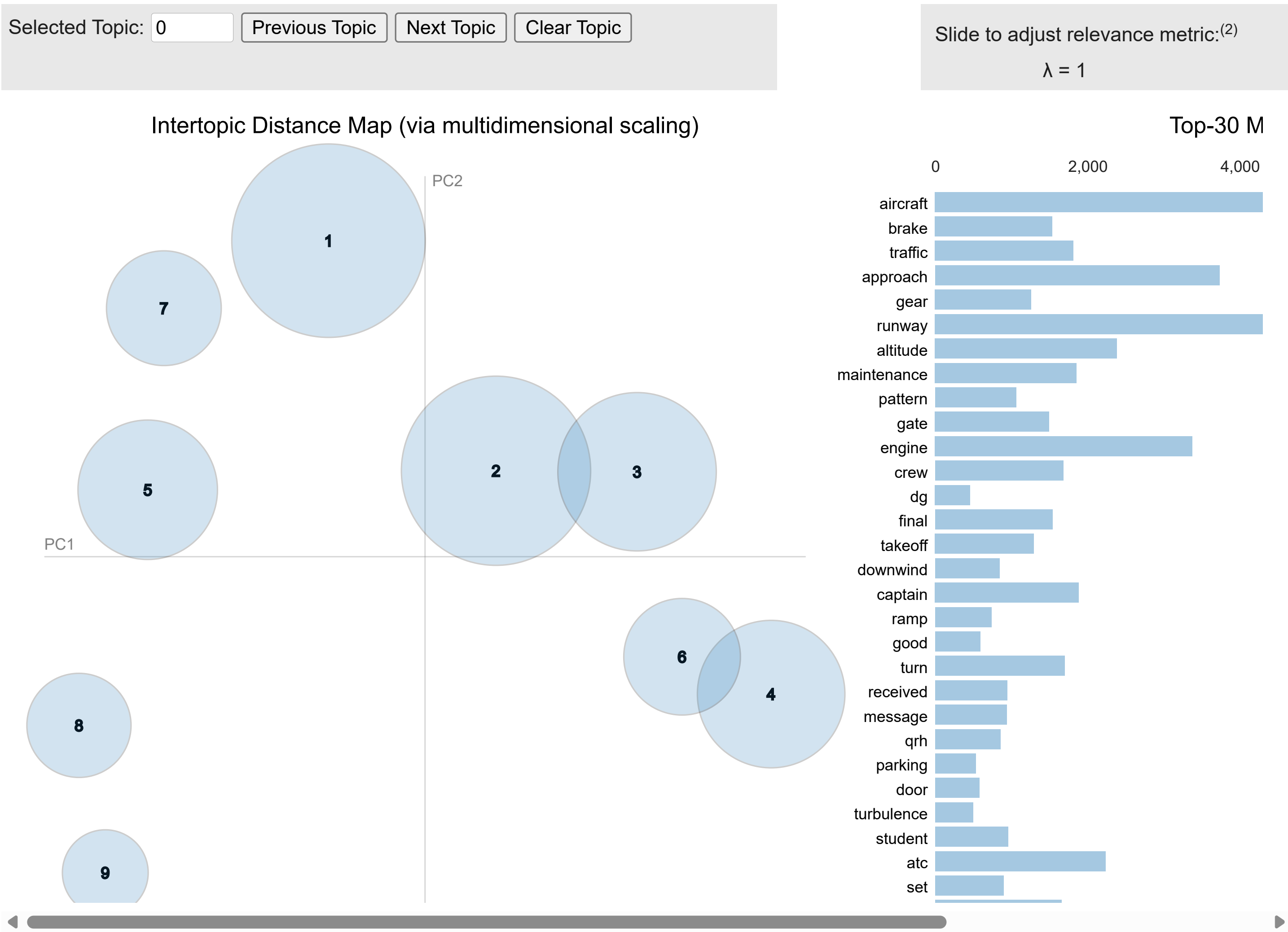
Fewer topics (5–6) or too many (19–20) result in lower coherence, which suggests the model either underfits (not enough thematic separation) or overfits (too many fragmented topics).

In [32]:

```
# Fit final model using optimal number of topics (9)
final_lda_model = LdaMulticore(
    corpus=bow_gensim,
    id2word=dict_gensim,
    chunksize=2000,
    eta='auto',
    iterations=400,
    num_topics=9,
    passes=20,
```

```
    eval_every=None,
    random_state=42
)

# Visualize with pyLDAvis
import pyLDAvis.gensim_models
import pyLDAvis

pyLDAvis.enable_notebook()
lda_display = pyLDAvis.gensim_models.prepare(final_lda_model, bow_gensim, dict_gensim)
lda_display
```

Out[32]:

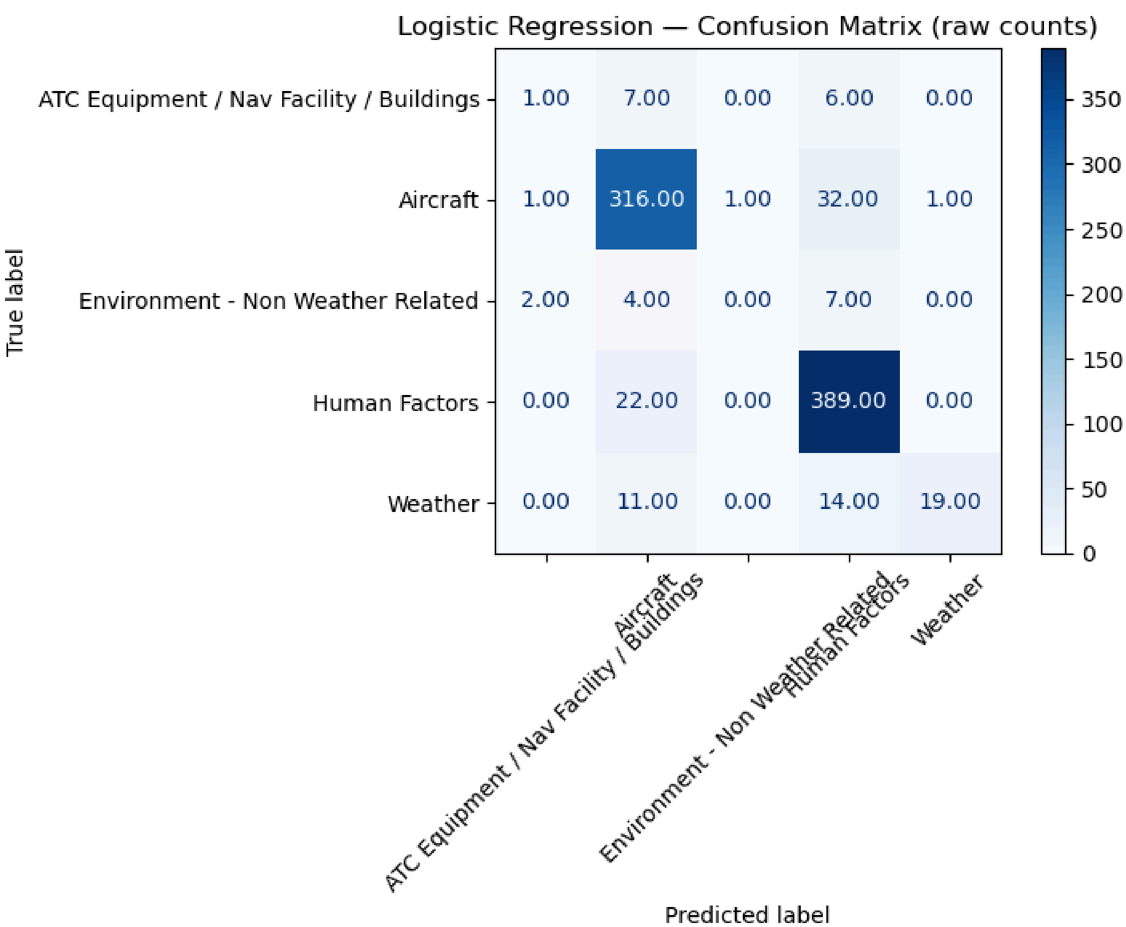


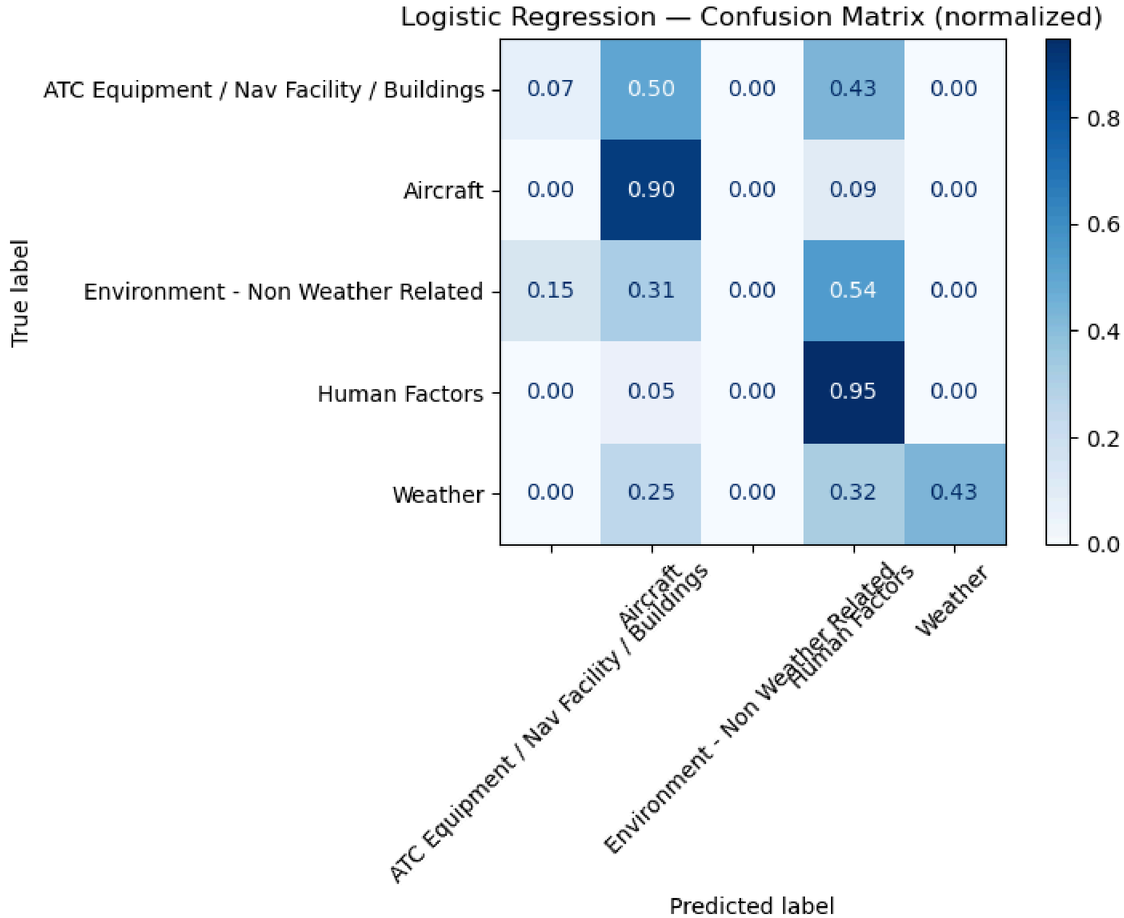
Confusion Matrix

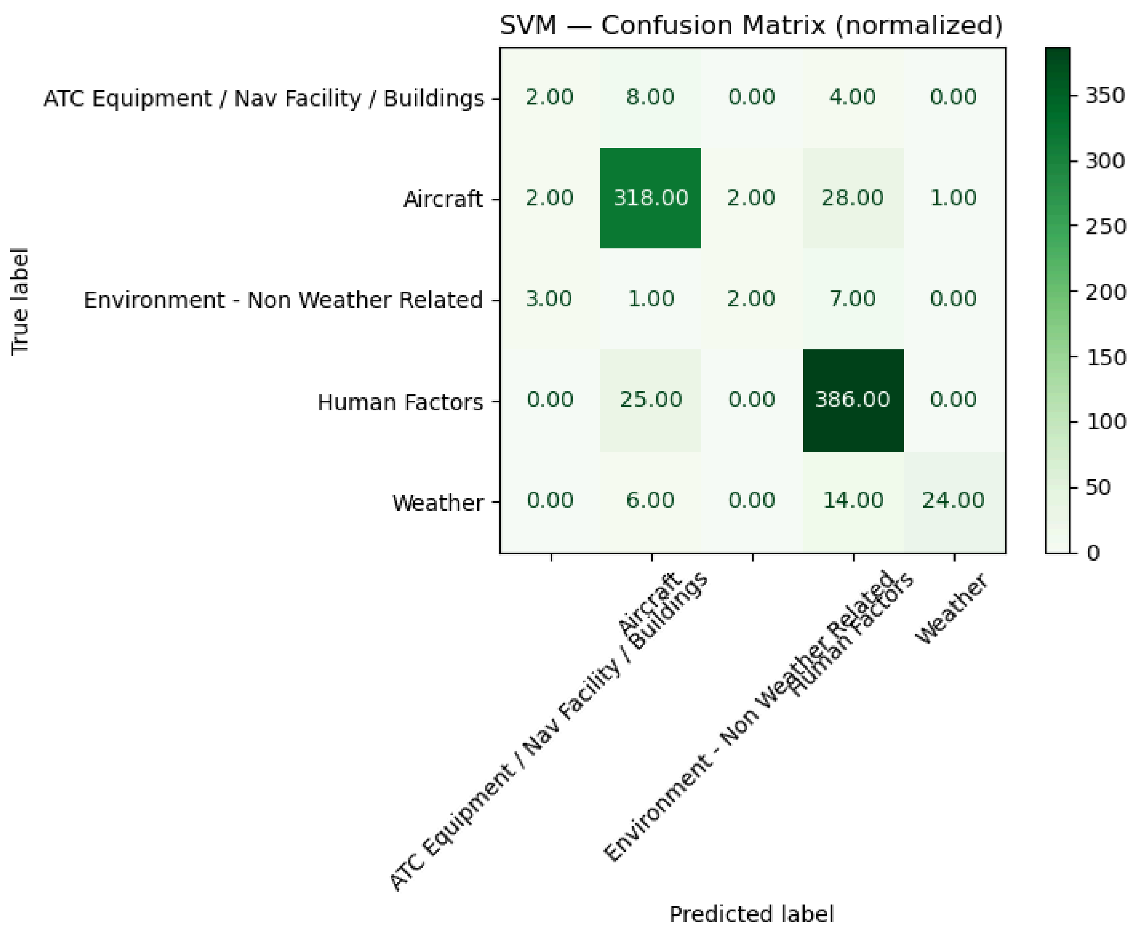
In [33]:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# --- Confusion matrix for Logistic Regression ---
cm_lr = confusion_matrix(y_test, y_pred, labels=lr.classes_)
disp_lr = ConfusionMatrixDisplay(
    confusion_matrix=cm_lr,
    display_labels=lr.classes_
)
fig, ax = plt.subplots(figsize=(8,6))
disp_lr.plot(
    include_values=True,
    cmap='Blues',
    ax=ax,
    xticks_rotation=45,
    values_format='.2f'
)
ax.set_title("Logistic Regression — Confusion Matrix (raw counts)")
plt.tight_layout()
plt.show()
```

```
# normalize by row
cm_lr_norm = confusion_matrix(y_test, y_pred, labels=lr.classes_, normalize='true')
disp_lr_norm = ConfusionMatrixDisplay(
    confusion_matrix=cm_lr_norm,
    display_labels=lr.classes_
)
fig, ax = plt.subplots(figsize=(8,6))
disp_lr_norm.plot(
    include_values=True,
    cmap='Blues',
    ax=ax,
    xticks_rotation=45,
    values_format='.2f'
)
ax.set_title("Logistic Regression — Confusion Matrix (normalized)")
plt.tight_layout()
plt.show()


# --- Confusion matrix for SVM ---
cm_svm = confusion_matrix(y_test, y_pred_svm, labels=svm.classes_)
disp_svm = ConfusionMatrixDisplay(
    confusion_matrix=cm_svm,
    display_labels=svm.classes_
)
fig, ax = plt.subplots(figsize=(8,6))
disp_svm.plot(
    include_values=True,
    cmap='Greens',
    ax=ax,
    xticks_rotation=45,
    values_format='.2f'
)
ax.set_title("SVM — Confusion Matrix (normalized)")
plt.tight_layout()
plt.show()
```







Logistic Regression performs very well on high-volume classes like Aircraft and Human Factors, but struggles with rarer classes, particularly ATC and Environment. ts overall accuracy is high, but improvements are needed in distinguishing less frequent and similar-sounding categories.

SVM maintains high accuracy on core categories and shows slightly better recall for the underrepresented "Weather" class. It remains conservative yet more capable in correctly flagging complex or ambiguous reports.

Conclusions: Project demonstrated classification model accuracy greater than 85% accuracy. Aircraft and Human Factors incidents were reliably identified. Topic modeling uncovered relevant themes such as maintenance, runway operations, comunication failures, thereby setting the focus to help improve existing safety interventions and aid in establishing future safety interventions.