In [82]:
```
%pip install pandas
%pip install matplotlib
%pip install seaborn
%pip install scikit-learn
%pip install nltk
%pip install wordcloud
%pip install emoji
```

```
Requirement already satisfied: pandas in ./.venv/lib/python3.10/site-packa
ges (2.2.3)
Requirement already satisfied: numpy>=1.22.4 in ./.venv/lib/python3.10/sit
e-packages (from pandas) (2.2.5)
Requirement already satisfied: python-dateutil>=2.8.2 in ./.venv/lib/pytho
n3.10/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in ./.venv/lib/python3.10/sit
e-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in ./.venv/lib/python3.10/si
te-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in ./.venv/lib/python3.10/site-pac
kages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: matplotlib in ./.venv/lib/python3.10/site-p
ackages (3.10.1)
Requirement already satisfied: contourpy>=1.0.1 in ./.venv/lib/python3.10/
site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in ./.venv/lib/python3.10/sit
e-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in ./.venv/lib/python3.1
0/site-packages (from matplotlib) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in ./.venv/lib/python3.1
0/site-packages (from matplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in ./.venv/lib/python3.10/site-
packages (from matplotlib) (2.2.5)
Requirement already satisfied: packaging>=20.0 in ./.venv/lib/python3.10/s
ite-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in ./.venv/lib/python3.10/site-pa
ckages (from matplotlib) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in ./.venv/lib/python3.10/
site-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in ./.venv/lib/python
3.10/site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in ./.venv/lib/python3.10/site-pac
kages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: seaborn in ./.venv/lib/python3.10/site-pack
ages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in ./.venv/lib/python
3.10/site-packages (from seaborn) (2.2.5)
Requirement already satisfied: pandas>=1.2 in ./.venv/lib/python3.10/site-
packages (from seaborn) (2.2.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in ./.venv/lib/pyth
on3.10/site-packages (from seaborn) (3.10.1)
Requirement already satisfied: contourpy>=1.0.1 in ./.venv/lib/python3.10/
site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.2)
Requirement already satisfied: cycler>=0.10 in ./.venv/lib/python3.10/sit
e-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in ./.venv/lib/python3.1
0/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in ./.venv/lib/python3.1
0/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in ./.venv/lib/python3.10/s
ite-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (25.0)
Requirement already satisfied: pillow>=8 in ./.venv/lib/python3.10/site-pa
ckages (from matplotlib!=3.6.1,>=3.4->seaborn) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in ./.venv/lib/python3.10/
site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in ./.venv/lib/python
3.10/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
```

```
Requirement already satisfied: pytz>=2020.1 in ./.venv/lib/python3.10/sit
e-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in ./.venv/lib/python3.10/si
te-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in ./.venv/lib/python3.10/site-pac
kages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.1
7.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: scikit-learn in ./.venv/lib/python3.10/sit
e-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in ./.venv/lib/python3.10/sit
e-packages (from scikit-learn) (2.2.5)
Requirement already satisfied: scipy>=1.6.0 in ./.venv/lib/python3.10/sit
e-packages (from scikit-learn) (1.15.2)
Requirement already satisfied: joblib>=1.2.0 in ./.venv/lib/python3.10/sit
e-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in ./.venv/lib/python
3.10/site-packages (from scikit-learn) (3.6.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: nltk in ./.venv/lib/python3.10/site-package
s (3.9.1)
Requirement already satisfied: click in ./.venv/lib/python3.10/site-packag
es (from nltk) (8.1.8)
Requirement already satisfied: joblib in ./.venv/lib/python3.10/site-packa
ges (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in ./.venv/lib/python3.10/s
ite-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in ./.venv/lib/python3.10/site-package
s (from nltk) (4.67.1)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: wordcloud in ./.venv/lib/python3.10/site-pa
ckages (1.9.4)
Requirement already satisfied: numpy>=1.6.1 in ./.venv/lib/python3.10/sit
e-packages (from wordcloud) (2.2.5)
Requirement already satisfied: pillow in ./.venv/lib/python3.10/site-packa
ges (from wordcloud) (11.2.1)
Requirement already satisfied: matplotlib in ./.venv/lib/python3.10/site-p
ackages (from wordcloud) (3.10.1)
Requirement already satisfied: contourpy>=1.0.1 in ./.venv/lib/python3.10/
site-packages (from matplotlib->wordcloud) (1.3.2)
Requirement already satisfied: cycler>=0.10 in ./.venv/lib/python3.10/sit
e-packages (from matplotlib->wordcloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in ./.venv/lib/python3.1
0/site-packages (from matplotlib->wordcloud) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in ./.venv/lib/python3.1
0/site-packages (from matplotlib->wordcloud) (1.4.8)
Requirement already satisfied: packaging>=20.0 in ./.venv/lib/python3.10/s
ite-packages (from matplotlib->wordcloud) (25.0)
Requirement already satisfied: pyparsing>=2.3.1 in ./.venv/lib/python3.10/
site-packages (from matplotlib->wordcloud) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in ./.venv/lib/python
3.10/site-packages (from matplotlib->wordcloud) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in ./.venv/lib/python3.10/site-pac
kages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: emoji in ./.venv/lib/python3.10/site-packag
es (2.14.1)
Note: you may need to restart the kernel to use updated packages.
```

In [83]:

```python
import matplotlib.pyplot as plt
```

```
import seaborn as sns
import numpy as np

import pandas as pd
df = pd.read_csv("scitweets_export.tsv", sep="\t")
df.head()
```

Out[83]:

	Unnamed: 0	tweet_id	text	science_related	scientific_claim	scie
0	0	316669998137483264	Knees are a bit sore. i guess that's a sign th...	0	0.0	
1	1	319090866545385472	McDonald's breakfast stop then the gym 🏀 💪	0	0.0	
2	2	322030931022065664	Can any Gynecologist with Cancer Experience ex...	1	1.0	
3	3	322694830620807168	Couch-lock highs lead to sleeping in the couch...	1	1.0	
4	4	328524426658328576	Does daily routine help prevent problems with ...	1	1.0	

In [122…

```
# Remove warning messages
import warnings

from sklearn.exceptions import ConvergenceWarning, UndefinedMetricWarning
warnings.filterwarnings("ignore", category=UserWarning, module="matplotli
warnings.filterwarnings("ignore", category=UserWarning, module="seaborn")
warnings.filterwarnings("ignore", category=UserWarning, module="pandas")
warnings.filterwarnings("ignore", category=UserWarning, module="wordcloud
warnings.filterwarnings("ignore", category=FutureWarning, module="pandas"
warnings.filterwarnings("ignore", category=DeprecationWarning, module="pa
warnings.filterwarnings("ignore", category=ConvergenceWarning, module="sk
warnings.filterwarnings("ignore", category=UndefinedMetricWarning, module
```

In [85]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X_all = vectorizer.fit_transform(df['text'])
```

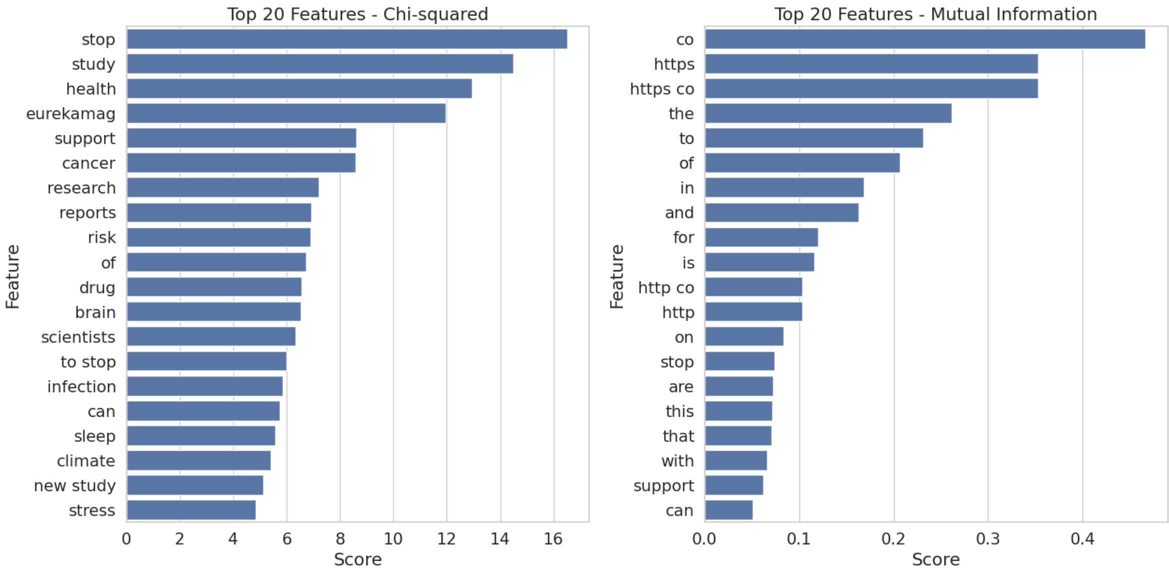
In [119…

```
# Compare different feature extraction methods
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_selection import SelectKBest, chi2, mutual_info_clas

# Create label for Task 1
y_task1 = df['science_related']
```

```
# 1. Count Vectorizer
count_vec = CountVectorizer(max_features=5000)
X_count = count_vec.fit_transform(df['text'])

# 2. TF-IDF with more parameters
tfidf_vec = TfidfVectorizer(max_features=5000,
                           min_df=5,
                           max_df=0.8,
                           ngram_range=(1, 2))
X_tfidf = tfidf_vec.fit_transform(df['text'])

# 3. TF-IDF with preprocessing already done
tfidf_processed = TfidfVectorizer(max_features=5000)
X_tfidf_processed = tfidf_processed.fit_transform(df['text'])

# Compare feature extraction methods
print(f"Count Vectorizer Features: {X_count.shape}")
print(f"TF-IDF Vectorizer Features: {X_tfidf.shape}")
print(f"TF-IDF on Preprocessed Text Features: {X_tfidf_processed.shape}")

# Feature selection using Chi-squared
selector_chi2 = SelectKBest(chi2, k=100)
X_chi2 = selector_chi2.fit_transform(X_tfidf, y_task1)

# Feature selection using Mutual Information
selector_mi = SelectKBest(mutual_info_classif, k=100)
X_mi = selector_mi.fit_transform(X_tfidf, y_task1)

print(f"\nFeatures after Chi-squared selection: {X_chi2.shape}")
print(f"Features after Mutual Information selection: {X_mi.shape}")

# Get and visualize the most important features
chi2_selected_indices = selector_chi2.get_support(indices=True)
mi_selected_indices = selector_mi.get_support(indices=True)

chi2_feature_names = np.array(tfidf_vec.get_feature_names_out())[chi2_sel
mi_feature_names = np.array(tfidf_vec.get_feature_names_out())[mi_selecte

# Plot top 20 features by importance
plt.figure(figsize=(16, 8))

plt.subplot(1, 2, 1)
chi2_scores = selector_chi2.scores_[chi2_selected_indices]
chi2_features_df = pd.DataFrame({'Feature': chi2_feature_names, 'Score':
chi2_features_df = chi2_features_df.sort_values('Score', ascending=False)
sns.barplot(x='Score', y='Feature', data=chi2_features_df)
plt.title('Top 20 Features - Chi-squared')

plt.subplot(1, 2, 2)
mi_scores = selector_mi.scores_[mi_selected_indices]
mi_features_df = pd.DataFrame({'Feature': mi_feature_names, 'Score': mi_s
mi_features_df = mi_features_df.sort_values('Score', ascending=False).hea
sns.barplot(x='Score', y='Feature', data=mi_features_df)
plt.title('Top 20 Features - Mutual Information')

plt.tight_layout()
plt.show()

# We'll use the TF-IDF on preprocessed text for subsequent modeling
```

```
X_selected = X_tfidf
```

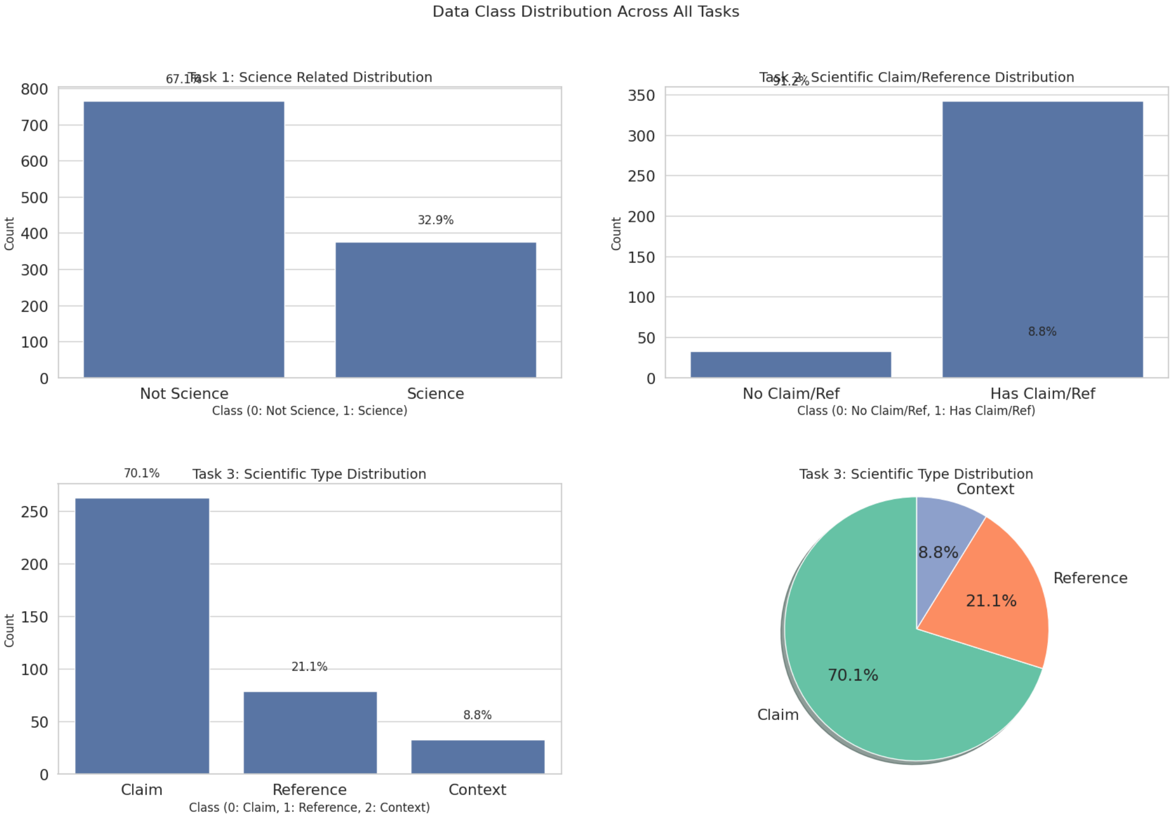
```
Count Vectorizer Features: (1140, 5000)
TF-IDF Vectorizer Features: (1140, 693)
TF-IDF on Preprocessed Text Features: (1140, 5000)

Features after Chi-squared selection: (1140, 100)
Features after Mutual Information selection: (1140, 100)
```



In [133…
```
# Create labels for all tasks
df['task1_label'] = df['science_related']

df_sci = df[df['science_related'] == 1].copy()
df_sci['task2_label'] = ((df_sci['scientific_claim'] == 1.0) | (df_sci['s
df_sci['task3_label'] = df_sci[['scientific_claim', 'scientific_reference
df_sci['task3_label'] = df_sci['task3_label'].map({
    'scientific_claim': 0,
    'scientific_reference': 1,
    'scientific_context': 2
})
```

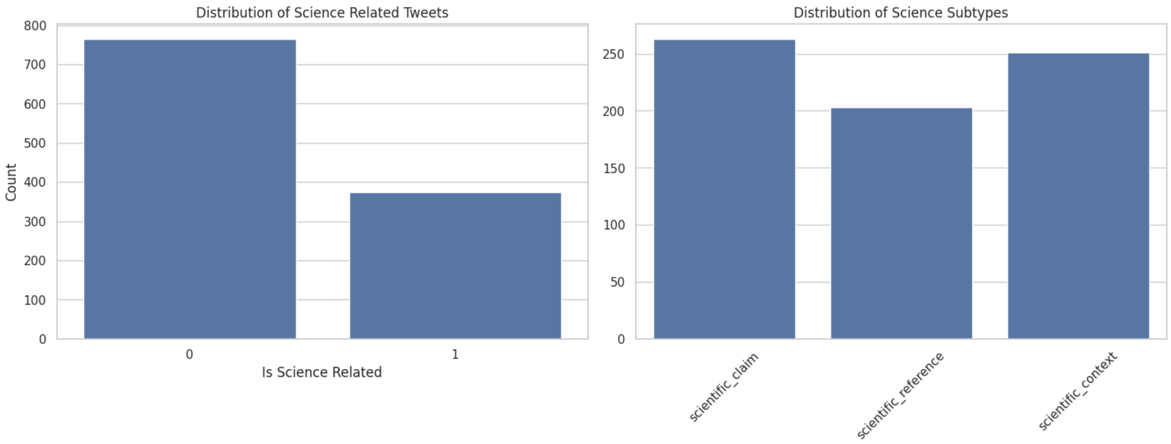
In [134…
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

# Set up figure for all three tasks
plt.figure(figsize=(18, 12))

# Task 1: Science Related Classification
plt.subplot(2, 2, 1)
task1_counts = df['task1_label'].value_counts()
sns.barplot(x=task1_counts.index, y=task1_counts.values)
plt.title('Task 1: Science Related Distribution', fontsize=14)
plt.xlabel('Class (0: Not Science, 1: Science)', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks([0, 1], ['Not Science', 'Science'])

# Add percentage labels on the bars
total = sum(task1_counts)
for i, count in enumerate(task1_counts):
    percentage = count / total * 100
    plt.text(i, count + 50, f'{percentage:.1f}%', ha='center', fontsize=1
```

```python
# Task 2: Scientific Claim/Reference Classification
plt.subplot(2, 2, 2)
task2_counts = df_sci['task2_label'].value_counts()
sns.barplot(x=task2_counts.index, y=task2_counts.values)
plt.title('Task 2: Scientific Claim/Reference Distribution', fontsize=14)
plt.xlabel('Class (0: No Claim/Ref, 1: Has Claim/Ref)', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks([0, 1], ['No Claim/Ref', 'Has Claim/Ref'])

# Add percentage labels
total = sum(task2_counts)
for i, count in enumerate(task2_counts):
    percentage = count / total * 100
    plt.text(i, count + 20, f'{percentage:.1f}%', ha='center', fontsize=1

# Task 3: Scientific Type Classification
plt.subplot(2, 2, 3)
task3_counts = df_sci['task3_label'].value_counts().sort_index()
sns.barplot(x=task3_counts.index, y=task3_counts.values)
plt.title('Task 3: Scientific Type Distribution', fontsize=14)
plt.xlabel('Class (0: Claim, 1: Reference, 2: Context)', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks([0, 1, 2], ['Claim', 'Reference', 'Context'])

# Add percentage labels
total = sum(task3_counts)
for i, count in enumerate(task3_counts):
    percentage = count / total * 100
    plt.text(i, count + 20, f'{percentage:.1f}%', ha='center', fontsize=1

# Pie chart for Task 3 (multiclass is better visualized as pie)
plt.subplot(2, 2, 4)
labels = ['Claim', 'Reference', 'Context']
plt.pie(task3_counts, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90, colors=sns.color_palette('Set2'))
plt.axis('equal')
plt.title('Task 3: Scientific Type Distribution', fontsize=14)

plt.tight_layout(pad=3.0)
plt.suptitle('Data Class Distribution Across All Tasks', fontsize=16, y=1
plt.show()
```



In [146…

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Set style
sns.set(style="whitegrid")
plt.figure(figsize=(15, 10))

# Plot distribution of science_related tweets
plt.subplot(2, 2, 1)
sns.countplot(x='science_related', data=df)
plt.title('Distribution of Science Related Tweets')
plt.xlabel('Is Science Related')
plt.ylabel('Count')

# Plot distribution of science subtypes for science-related tweets
sci_df = df[df['science_related'] == 1]
plt.subplot(2, 2, 2)
subtypes = ['scientific_claim', 'scientific_reference', 'scientific_conte
sns.barplot(x=subtypes, y=[sci_df[col].sum() for col in subtypes])
plt.title('Distribution of Science Subtypes')
plt.xticks(rotation=45)
plt.tight_layout()


plt.tight_layout()
plt.show()
```



In [136…

```
# More data analysis and visualizations

# 1. Compare text characteristics by class
plt.figure(figsize=(20, 15))
plt.suptitle("Advanced Data Analysis", fontsize=20, y=0.95)

# Plot 1: Text length distribution by class and prediction correctness
plt.subplot(2, 2, 1)
df['text_length'] = df['text'].apply(len)
df['processed_text_length'] = df['processed_text'].apply(len)

# Create a column indicating if prediction was correct for samples in tes
mask_test = df.index.isin(y_test_task1.index)
df_test = df[mask_test].copy()
df_test['prediction'] = y_pred
df_test['correct'] = df_test['prediction'] == df_test['task1_label']

# Plot text length distribution by correct/incorrect predictions
sns.boxplot(x='task1_label', y='text_length', hue='correct', data=df_test
plt.title('Text Length Distribution by Class and Prediction Correctness')
plt.xlabel('Class (0: Not Science, 1: Science)')
plt.ylabel('Text Length')
plt.legend(title='Correct Prediction')

# Plot 2: Character usage analysis
plt.subplot(2, 2, 2)
df['has_url'] = df['text'].str.contains('http|www', regex=True)
df['has_hashtag'] = df['text'].str.contains('#', regex=True)
df['has_mention'] = df['text'].str.contains('@', regex=True)
df['has_number'] = df['text'].str.contains('\d', regex=True)
df['has_emoji'] = df['text'].apply(lambda x: emoji.emoji_count(x) > 0)

char_features = ['has_url', 'has_hashtag', 'has_mention', 'has_number', '
char_usage = pd.DataFrame({
    'Feature': [],
    'Class': [],
    'Percentage': []
})

for feature in char_features:
    for label in [0, 1]:
        subset = df[df['task1_label'] == label]
        percentage = subset[feature].mean() * 100
        char_usage = pd.concat([char_usage, pd.DataFrame({
            'Feature': [feature.replace('has_', '')],
```

```
            'Class': ['Not Science' if label == 0 else 'Science'],
            'Percentage': [percentage]
        })], ignore_index=True)

sns.barplot(x='Feature', y='Percentage', hue='Class', data=char_usage)
plt.title('Character Usage by Class')
plt.ylabel('Percentage of Tweets')
plt.ylim(0, 100)
plt.legend(title='Class')

# Plot 3: Word count distribution by class
plt.subplot(2, 2, 3)
df['word_count'] = df['text'].apply(lambda x: len(x.split()))
df['processed_word_count'] = df['processed_text'].apply(lambda x: len(x.s

sns.violinplot(x='task1_label', y='word_count', hue='science_related', da
plt.title('Word Count Distribution by Class')
plt.xlabel('Class (0: Not Science, 1: Science)')
plt.ylabel('Word Count')

# Plot 4: Confusion matrices for all tasks
plt.subplot(2, 2, 4)

# Function to create a normalized confusion matrix
def plot_cm(cm, labels, title, ax):
    cm_norm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    sns.heatmap(cm_norm, annot=True, fmt='.2f', cmap='Blues',
                xticklabels=labels, yticklabels=labels, ax=ax)
    ax.set_title(title)
    ax.set_ylabel('True Label')
    ax.set_xlabel('Predicted Label')


# Create a word usage comparison visualization
plt.figure(figsize=(16, 8))

# Create a mask for correctly and incorrectly classified samples
correct_mask = df_test['correct'] == True
incorrect_mask = df_test['correct'] == False

# Get top words in correctly classified science tweets vs incorrectly cla
if sum(correct_mask & (df_test['task1_label'] == 1)) > 0 and sum(incorrec
    correct_science_text = ' '.join(df_test[correct_mask & (df_test['task
    incorrect_science_text = ' '.join(df_test[incorrect_mask & (df_test['

    # Create word clouds for comparison
    plt.subplot(1, 2, 1)
    wordcloud_correct = WordCloud(width=800, height=400, background_color
                                  max_words=50, colormap='viridis').genera
    plt.imshow(wordcloud_correct, interpolation='bilinear')
    plt.axis('off')
    plt.title('Words in Correctly Classified Science Tweets')

    plt.subplot(1, 2, 2)
    wordcloud_incorrect = WordCloud(width=800, height=400, background_col
                                    max_words=50, colormap='plasma').gener
    plt.imshow(wordcloud_incorrect, interpolation='bilinear')
    plt.axis('off')
    plt.title('Words in Incorrectly Classified Science Tweets')
```

```
    plt.tight_layout()
    plt.show()

# Summary statistics table
stats_df = pd.DataFrame({
    'Metric': ['Average Text Length', 'Average Word Count', 'URLs (%)', '
    'Not Science': [
        df[df['task1_label'] == 0]['text_length'].mean(),
        df[df['task1_label'] == 0]['word_count'].mean(),
        df[df['task1_label'] == 0]['has_url'].mean() * 100,
        df[df['task1_label'] == 0]['has_hashtag'].mean() * 100,
        df[df['task1_label'] == 0]['has_mention'].mean() * 100
    ],
    'Science': [
        df[df['task1_label'] == 1]['text_length'].mean(),
        df[df['task1_label'] == 1]['word_count'].mean(),
        df[df['task1_label'] == 1]['has_url'].mean() * 100,
        df[df['task1_label'] == 1]['has_hashtag'].mean() * 100,
        df[df['task1_label'] == 1]['has_mention'].mean() * 100
    ]
})

# Display the table with a visualization
plt.figure(figsize=(12, 6))
ax = plt.subplot(111, frame_on=False)
ax.xaxis.set_visible(False)
ax.yaxis.set_visible(False)

table = plt.table(cellText=stats_df.values,
                  colLabels=stats_df.columns,
                  cellLoc='center',
                  loc='center',
                  bbox=[0.2, 0.2, 0.6, 0.5])
table.auto_set_font_size(False)
table.set_fontsize(12)
table.scale(1.2, 1.5)
plt.title('Summary Statistics by Class', fontsize=16, pad=20)
plt.show()
```

Summary Statistics by Class

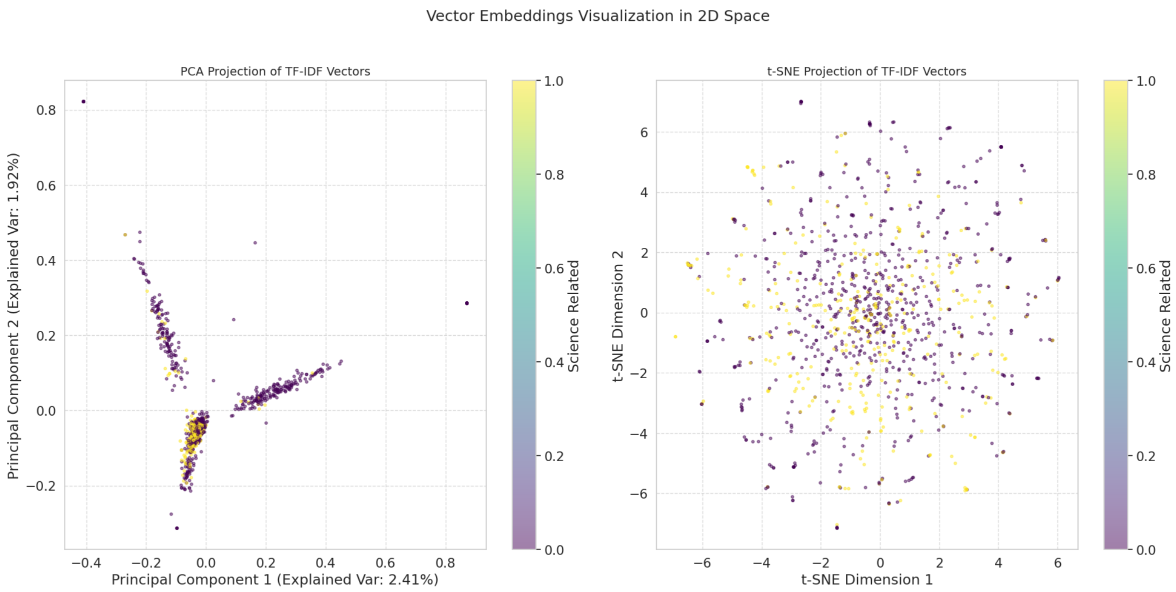
Metric	Not Science	Science
Average Text Length	130.96993464052287	149.336
Average Word Count	18.870588235294118	19.952
URLs (%)	56.07843137254902	80.80000000000001
Hashtags (%)	32.549019607843135	32.800000000000004
Mentions (%)	37.908496732026144	25.6

In [139…

```python
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
```

```
import matplotlib.pyplot as plt
import numpy as np

# Create a figure to compare different dimensionality reduction technique
plt.figure(figsize=(20, 10))

# Get class labels for coloring
labels = df['science_related'].values

# 1. PCA - Fast but linear projection
plt.subplot(1, 2, 1)
pca = PCA(n_components=2, random_state=42)
X_pca = pca.fit_transform(X_selected.toarray())

# Create scatter plot
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels,
            cmap='viridis', alpha=0.5, s=10)
plt.title(f'PCA Projection of TF-IDF Vectors', fontsize=14)
plt.xlabel(f'Principal Component 1 (Explained Var: {pca.explained_varianc
plt.ylabel(f'Principal Component 2 (Explained Var: {pca.explained_varianc
plt.colorbar(label='Science Related')
plt.grid(True, linestyle='--', alpha=0.7)

# 2. t-SNE - Better for preserving local structure (slower)
plt.subplot(1, 2, 2)
# Use a subset of data for t-SNE as it's computationally intensive
sample_indices = np.random.choice(X_selected.shape[0], min(3000, X_select
X_sample = X_selected[sample_indices].toarray()
labels_sample = labels[sample_indices]

# Apply t-SNE
tsne = TSNE(n_components=2, random_state=42, perplexity=30, n_iter=1000)
X_tsne = tsne.fit_transform(X_sample)

# Create scatter plot
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=labels_sample,
            cmap='viridis', alpha=0.5, s=10)
plt.title('t-SNE Projection of TF-IDF Vectors', fontsize=14)
plt.xlabel('t-SNE Dimension 1')
plt.ylabel('t-SNE Dimension 2')
plt.colorbar(label='Science Related')
plt.grid(True, linestyle='--', alpha=0.7)

plt.suptitle('Vector Embeddings Visualization in 2D Space', fontsize=18)
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

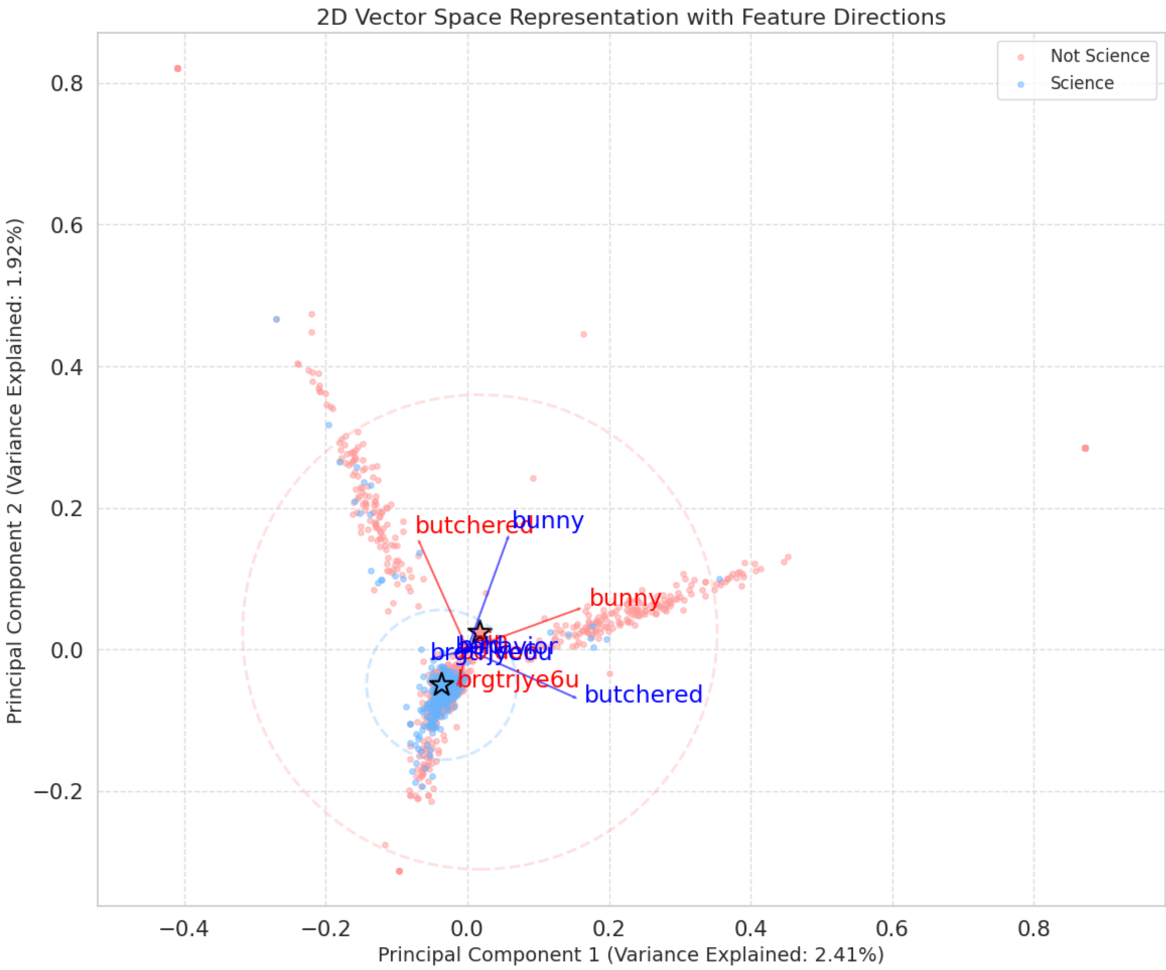
```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/manifold/_t_sne.py:1164: FutureWarning: 'n_iter' was renamed to 'max
_iter' in version 1.5 and will be removed in 1.7.
  warnings.warn(
```



In [140…

```python
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np
from sklearn.decomposition import PCA

# Get the dense version of features
X_dense = X_selected.toarray()

# Apply PCA
pca = PCA(n_components=2)
X_2d = pca.fit_transform(X_dense)

# Create a figure
plt.figure(figsize=(12, 10))

# Define colors for each class
colors = ['#ff9999', '#66b3ff']
labels_unique = [0, 1]
label_names = ['Not Science', 'Science']

# Plot points by class
for i, label in enumerate(labels_unique):
    # Get indices for this class
    indices = df['science_related'] == label

    # Plot points for this class
    plt.scatter(X_2d[indices, 0], X_2d[indices, 1],
                c=colors[i], alpha=0.5, s=15,
                label=label_names[i])

    # Calculate and plot centroid
    centroid = X_2d[indices].mean(axis=0)
    plt.scatter(centroid[0], centroid[1],
                marker='*', s=300, c=colors[i],
                edgecolor='black', linewidth=1.5)

    # Draw a circle around majority of points in this class
    std_dev = X_2d[indices].std(axis=0).mean() * 2  # 2 std dev circle
    circle = plt.Circle((centroid[0], centroid[1]), std_dev,
                        color=colors[i], fill=False,
                        linestyle='--', linewidth=2, alpha=0.3)
```

```
    plt.gca().add_patch(circle)

# Add vector directions of top features
feature_names = vectorizer.get_feature_names_out()
pca_components = pca.components_

# Get top influential features in each principal component
n_top_features = 10
sorted_idx = np.argsort(-np.abs(pca_components), axis=1)[:, :n_top_featur

# Scale for the arrows
scale = np.abs(X_2d).max() * 0.2

# Plot feature vectors
for i, (idx, c) in enumerate(zip(sorted_idx, ['red', 'blue'])):
    for j, feature_idx in enumerate(idx):
        feature_name = feature_names[feature_idx]
        # Only plot first few to avoid clutter
        if j < 5:
            plt.arrow(0, 0,
                      pca_components[i, feature_idx] * scale,
                      pca_components[(i+1)%2, feature_idx] * scale,
                      color=c, alpha=0.5)
            plt.text(pca_components[i, feature_idx] * scale * 1.1,
                     pca_components[(i+1)%2, feature_idx] * scale * 1.1,
                     feature_name, color=c)

plt.title('2D Vector Space Representation with Feature Directions', fonts
plt.xlabel(f'Principal Component 1 (Variance Explained: {pca.explained_va
plt.ylabel(f'Principal Component 2 (Variance Explained: {pca.explained_va
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=12)
plt.axis('equal')  # Equal scaling for x and y
plt.tight_layout()
plt.show()
```



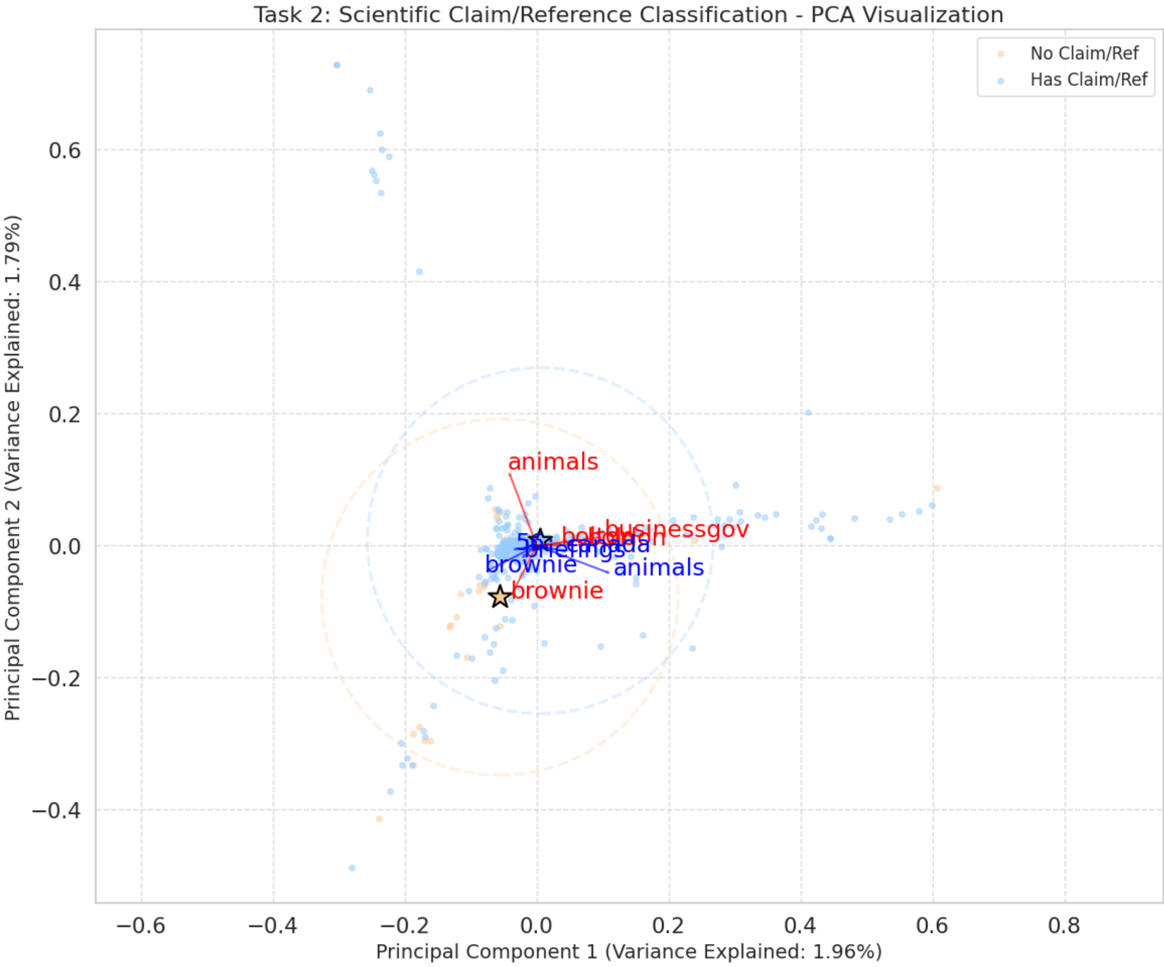
In [142…

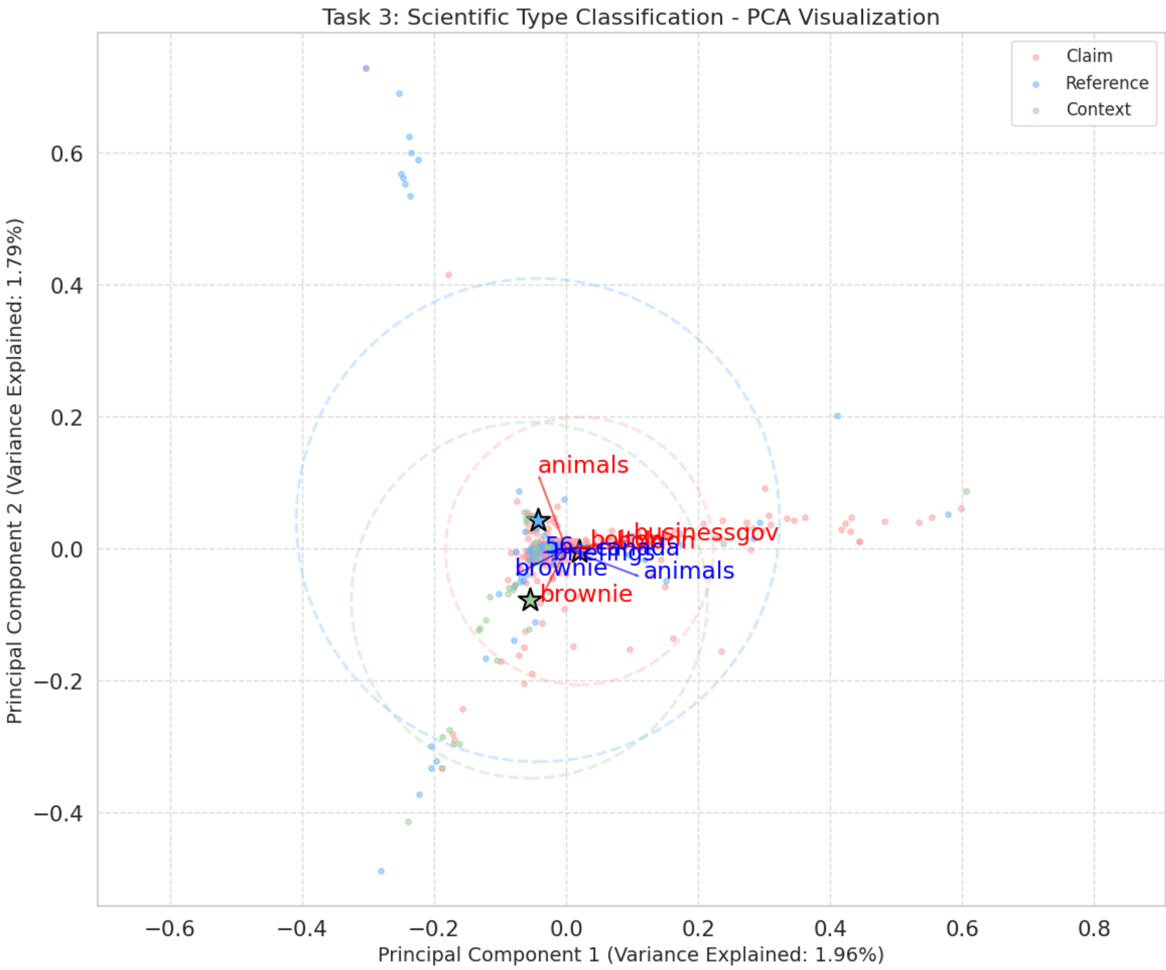
```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np
from sklearn.decomposition import PCA

# First, we need to get features only for science-related tweets
# Get the dense version of features for science-related tweets
X_dense_sci = X_selected[(df['science_related'] == 1).values].toarray()

# Get the corresponding labels for task2 and task3
y_task2 = df_sci['task2_label']
y_task3 = df_sci['task3_label']

# ================= TASK 2 VISUALIZATION =================
# Apply PCA
pca_task2 = PCA(n_components=2)
X_2d_task2 = pca_task2.fit_transform(X_dense_sci)

# Create a figure
plt.figure(figsize=(12, 10))

# Define colors for each class in Task 2
colors_task2 = ['#ffcc99', '#99ccff']
labels_unique_task2 = [0, 1]
label_names_task2 = ['No Claim/Ref', 'Has Claim/Ref']

# Plot points by class
for i, label in enumerate(labels_unique_task2):
    # Get indices for this class
    indices = y_task2 == label
```

```
    # Plot points for this class
    plt.scatter(X_2d_task2[indices, 0], X_2d_task2[indices, 1],
                c=colors_task2[i], alpha=0.5, s=15,
                label=label_names_task2[i])

    # Calculate and plot centroid
    centroid = X_2d_task2[indices].mean(axis=0)
    plt.scatter(centroid[0], centroid[1],
                marker='*', s=300, c=colors_task2[i],
                edgecolor='black', linewidth=1.5)

    # Draw a circle around majority of points in this class
    std_dev = X_2d_task2[indices].std(axis=0).mean() * 2  # 2 std dev cir
    circle = plt.Circle((centroid[0], centroid[1]), std_dev,
                        color=colors_task2[i], fill=False,
                        linestyle='--', linewidth=2, alpha=0.3)
    plt.gca().add_patch(circle)

# Add vector directions of top features
feature_names = vectorizer.get_feature_names_out()
pca_components = pca_task2.components_

# Get top influential features in each principal component
n_top_features = 10
sorted_idx = np.argsort(-np.abs(pca_components), axis=1)[:, :n_top_featur

# Scale for the arrows
scale = np.abs(X_2d_task2).max() * 0.2

# Plot feature vectors
for i, (idx, c) in enumerate(zip(sorted_idx, ['red', 'blue'])):
    for j, feature_idx in enumerate(idx):
        feature_name = feature_names[feature_idx]
        # Only plot first few to avoid clutter
        if j < 5:
            plt.arrow(0, 0,
                      pca_components[i, feature_idx] * scale,
                      pca_components[(i+1)%2, feature_idx] * scale,
                      color=c, alpha=0.5)
            plt.text(pca_components[i, feature_idx] * scale * 1.1,
                     pca_components[(i+1)%2, feature_idx] * scale * 1.1,
                     feature_name, color=c)

plt.title('Task 2: Scientific Claim/Reference Classification - PCA Visual
plt.xlabel(f'Principal Component 1 (Variance Explained: {pca_task2.explai
plt.ylabel(f'Principal Component 2 (Variance Explained: {pca_task2.explai
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=12)
plt.axis('equal')  # Equal scaling for x and y
plt.tight_layout()
plt.show()

# ================ TASK 3 VISUALIZATION ================
# Apply PCA
pca_task3 = PCA(n_components=2)
X_2d_task3 = pca_task3.fit_transform(X_dense_sci)

# Create a figure
plt.figure(figsize=(12, 10))
```

```
# Define colors for each class in Task 3
colors_task3 = ['#ff9999', '#66b3ff', '#99cc99']  # Red, Blue, Green
labels_unique_task3 = [0, 1, 2]
label_names_task3 = ['Claim', 'Reference', 'Context']

# Plot points by class
for i, label in enumerate(labels_unique_task3):
    # Get indices for this class
    indices = y_task3 == label

    # Plot points for this class
    plt.scatter(X_2d_task3[indices, 0], X_2d_task3[indices, 1],
                c=colors_task3[i], alpha=0.5, s=15,
                label=label_names_task3[i])

    # Calculate and plot centroid
    centroid = X_2d_task3[indices].mean(axis=0)
    plt.scatter(centroid[0], centroid[1],
                marker='*', s=300, c=colors_task3[i],
                edgecolor='black', linewidth=1.5)

    # Draw a circle around majority of points in this class
    std_dev = X_2d_task3[indices].std(axis=0).mean() * 2  # 2 std dev cir
    circle = plt.Circle((centroid[0], centroid[1]), std_dev,
                        color=colors_task3[i], fill=False,
                        linestyle='--', linewidth=2, alpha=0.3)
    plt.gca().add_patch(circle)

# Add vector directions of top features
feature_names = vectorizer.get_feature_names_out()
pca_components = pca_task3.components_

# Get top influential features in each principal component
n_top_features = 10
sorted_idx = np.argsort(-np.abs(pca_components), axis=1)[:, :n_top_featur

# Scale for the arrows
scale = np.abs(X_2d_task3).max() * 0.2

# Plot feature vectors (we still use 2 components in PCA)
for i, (idx, c) in enumerate(zip(sorted_idx, ['red', 'blue'])):
    for j, feature_idx in enumerate(idx):
        feature_name = feature_names[feature_idx]
        # Only plot first few to avoid clutter
        if j < 5:
            plt.arrow(0, 0,
                      pca_components[i, feature_idx] * scale,
                      pca_components[(i+1)%2, feature_idx] * scale,
                      color=c, alpha=0.5)
            plt.text(pca_components[i, feature_idx] * scale * 1.1,
                     pca_components[(i+1)%2, feature_idx] * scale * 1.1,
                     feature_name, color=c)

plt.title('Task 3: Scientific Type Classification - PCA Visualization', f
plt.xlabel(f'Principal Component 1 (Variance Explained: {pca_task3.explai
plt.ylabel(f'Principal Component 2 (Variance Explained: {pca_task3.explai
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=12)
plt.axis('equal')  # Equal scaling for x and y
```

```
plt.tight_layout()
plt.show()
```





In [88]:
```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, MultinomialNB, ComplementNB,
from sklearn.metrics import classification_report, confusion_matrix, accu
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier


# Function to evaluate model performance
def evaluate_model(model, X_train, X_test, y_train, y_test, model_name):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True)
    cm = confusion_matrix(y_test, y_pred)

    print(f"Model: {model_name}")
    print(f"Accuracy: {accuracy:.4f}")
    print(classification_report(y_test, y_pred))

    return {
        'model_name': model_name,
        'accuracy': accuracy,
        'report': report,
        'confusion_matrix': cm,
        'y_pred': y_pred
    }
```

```python
# Get a dense version of our features for Gaussian NB
X_dense = X_selected.toarray()

# Split data for Task 1
X_train_task1, X_test_task1, y_train_task1, y_test_task1 = train_test_spl
    X_dense, df['task1_label'], test_size=0.2, random_state=42
)

# Compare different NB variants for Task 1
nb_models = {
    'Gaussian NB': GaussianNB(),
    'Multinomial NB': MultinomialNB(),
    'Complement NB': ComplementNB(),
    'Bernoulli NB': BernoulliNB(),
    'KNN'    : KNeighborsClassifier(n_neighbors=5),
    'SVM'    : SVC(kernel='linear', C=1),
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Random Forest': RandomForestClassifier(n_estimators=100),
}

task1_results = {}
print("Task 1: Science Related Classification\n" + "="*40)
for name, model in nb_models.items():
    task1_results[name] = evaluate_model(
        model, X_train_task1, X_test_task1, y_train_task1, y_test_task1,
    )
    print("\n")
```

```
Task 1: Science Related Classification
========================================
Model: Gaussian NB
Accuracy: 0.6711
              precision    recall  f1-score   support

           0       0.79      0.66      0.72       146
           1       0.53      0.70      0.60        82

    accuracy                           0.67       228
   macro avg       0.66      0.68      0.66       228
weighted avg       0.70      0.67      0.68       228



Model: Multinomial NB
Accuracy: 0.7544
              precision    recall  f1-score   support

           0       0.74      0.96      0.83       146
           1       0.84      0.39      0.53        82

    accuracy                           0.75       228
   macro avg       0.79      0.67      0.68       228
weighted avg       0.77      0.75      0.73       228



Model: Complement NB
Accuracy: 0.7851
              precision    recall  f1-score   support

           0       0.84      0.82      0.83       146
           1       0.69      0.73      0.71        82

    accuracy                           0.79       228
   macro avg       0.77      0.77      0.77       228
weighted avg       0.79      0.79      0.79       228



Model: Bernoulli NB
Accuracy: 0.7895
              precision    recall  f1-score   support

           0       0.80      0.90      0.85       146
           1       0.77      0.59      0.67        82

    accuracy                           0.79       228
   macro avg       0.78      0.74      0.76       228
weighted avg       0.79      0.79      0.78       228



Model: KNN
Accuracy: 0.7105
              precision    recall  f1-score   support

           0       0.80      0.73      0.76       146
           1       0.58      0.68      0.63        82
```

```
    accuracy                           0.71       228
   macro avg       0.69      0.70      0.70       228
weighted avg       0.72      0.71      0.71       228



Model: SVM
Accuracy: 0.7895
              precision    recall  f1-score   support

           0       0.79      0.92      0.85       146
           1       0.79      0.56      0.66        82

    accuracy                           0.79       228
   macro avg       0.79      0.74      0.75       228
weighted avg       0.79      0.79      0.78       228



Model: Logistic Regression
Accuracy: 0.7675
              precision    recall  f1-score   support

           0       0.74      0.97      0.84       146
           1       0.89      0.40      0.55        82

    accuracy                           0.77       228
   macro avg       0.82      0.69      0.70       228
weighted avg       0.80      0.77      0.74       228



Model: Random Forest
Accuracy: 0.7544
              precision    recall  f1-score   support

           0       0.74      0.94      0.83       146
           1       0.80      0.43      0.56        82

    accuracy                           0.75       228
   macro avg       0.77      0.68      0.69       228
weighted avg       0.76      0.75      0.73       228
```

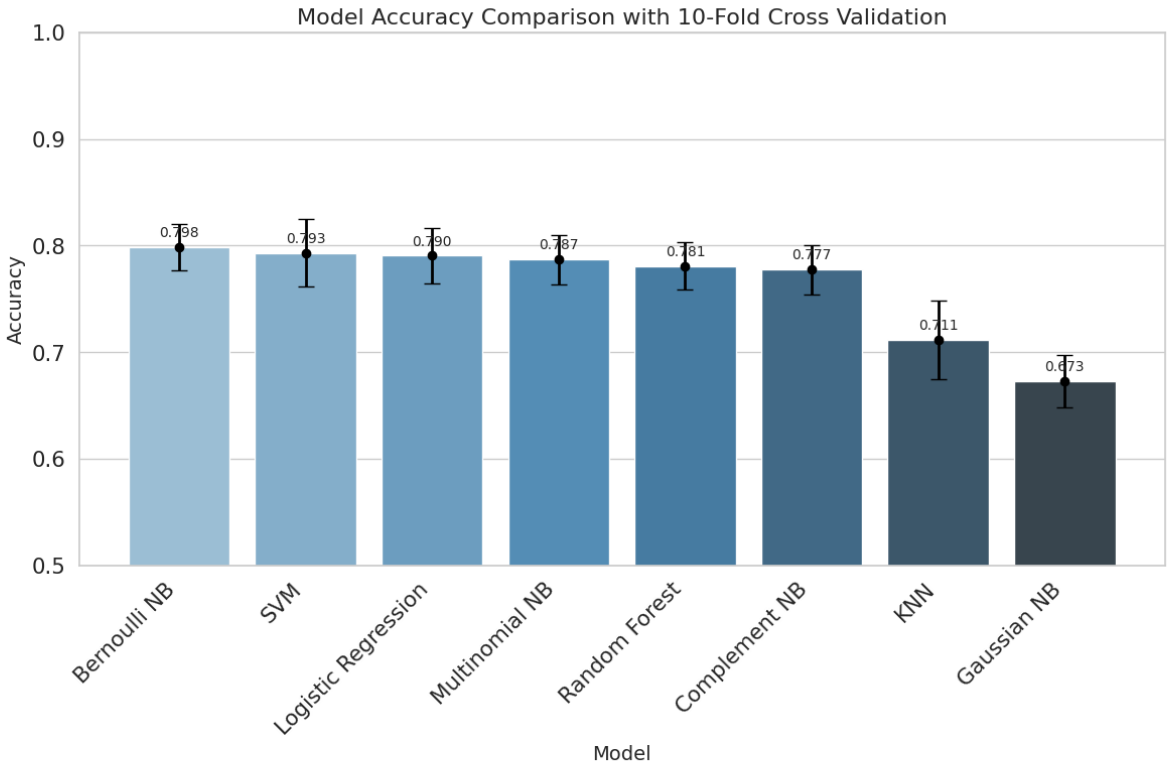
In [89]:
```python
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Initialize k-fold cross validation
k_folds = 10
skf = StratifiedKFold(n_splits=k_folds, shuffle=True, random_state=42)

# Dictionary to store results
cv_results = {}

# Perform k-fold cross validation for each model
for model_name, model in nb_models.items():
    print(f"Performing {k_folds}-fold cross-validation for {model_name}..
```

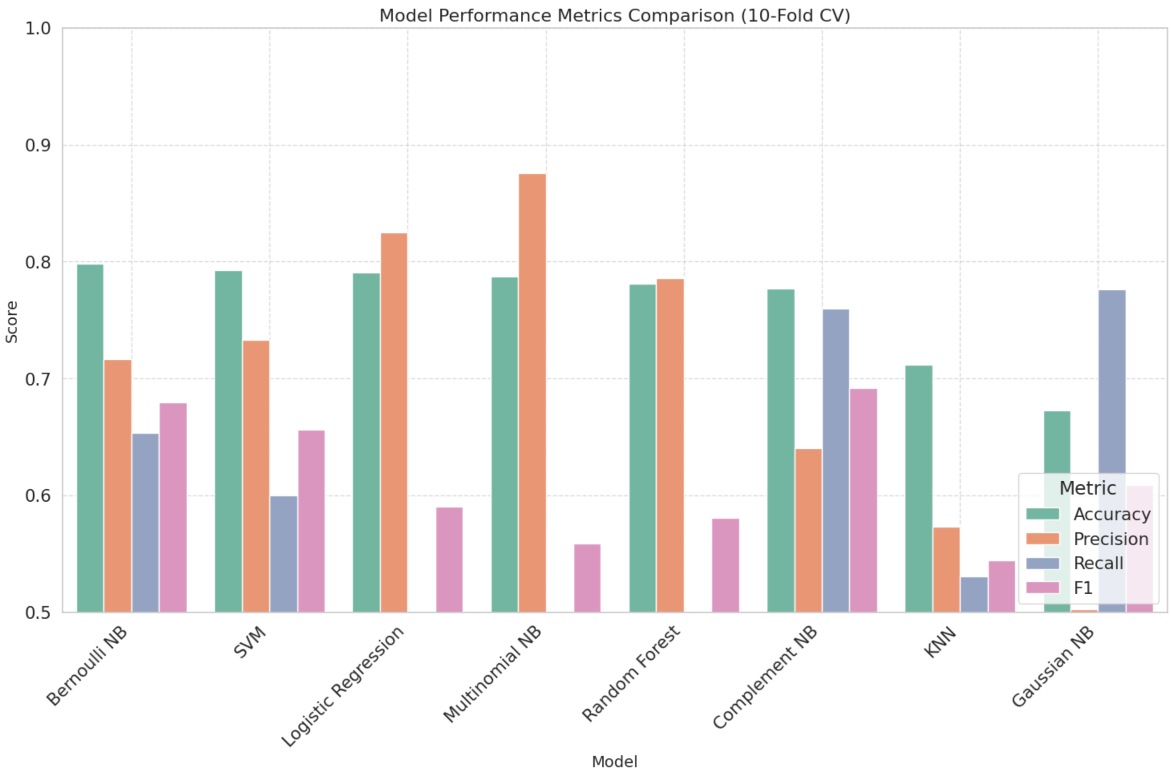
```
    # Initialize lists to store performance metrics for each fold
    fold_accuracy = []
    fold_precision = []
    fold_recall = []
    fold_f1 = []

    # For each fold
    for fold, (train_idx, test_idx) in enumerate(skf.split(X_dense, y_tas
        # Split data
        X_train_fold, X_test_fold = X_dense[train_idx], X_dense[test_idx]
        y_train_fold, y_test_fold = y_task1.iloc[train_idx], y_task1.iloc

        # Train model
        model.fit(X_train_fold, y_train_fold)

        # Make predictions
        y_pred_fold = model.predict(X_test_fold)

        # Calculate metrics
        acc = accuracy_score(y_test_fold, y_pred_fold)
        prec = precision_score(y_test_fold, y_pred_fold, zero_division=0)
        rec = recall_score(y_test_fold, y_pred_fold, zero_division=0)
        f1 = f1_score(y_test_fold, y_pred_fold, zero_division=0)

        fold_accuracy.append(acc)
        fold_precision.append(prec)
        fold_recall.append(rec)
        fold_f1.append(f1)

    # Store average metrics and standard deviations
    cv_results[model_name] = {
        'accuracy': {
            'mean': np.mean(fold_accuracy),
            'std': np.std(fold_accuracy)
        },
        'precision': {
            'mean': np.mean(fold_precision),
            'std': np.std(fold_precision)
        },
        'recall': {
            'mean': np.mean(fold_recall),
            'std': np.std(fold_recall)
        },
        'f1': {
            'mean': np.mean(fold_f1),
            'std': np.std(fold_f1)
        }
    }

    print(f"  Average: Accuracy={cv_results[model_name]['accuracy']['mean
    print(f"  Average: F1 Score={cv_results[model_name]['f1']['mean']:.4f
    print()

# Create DataFrame for visualization
results_df = pd.DataFrame({
    'Model': [],
    'Metric': [],
    'Mean': [],
    'Std': []
```

```
})

for model_name in cv_results:
    for metric in ['accuracy', 'precision', 'recall', 'f1']:
        results_df = pd.concat([results_df, pd.DataFrame({
            'Model': [model_name],
            'Metric': [metric.capitalize()],
            'Mean': [cv_results[model_name][metric]['mean']],
            'Std': [cv_results[model_name][metric]['std']]
        })], ignore_index=True)

# Sort models by accuracy
model_order = results_df[results_df['Metric'] == 'Accuracy'].sort_values(

# Create plots
plt.figure(figsize=(12, 8))
sns.set_style("whitegrid")

# Create bar plot for accuracy
ax = sns.barplot(
    data=results_df[results_df['Metric'] == 'Accuracy'],
    x='Model',
    y='Mean',
    order=model_order,
    palette='Blues_d'
)

# Add error bars
for i, model in enumerate(model_order):
    row = results_df[(results_df['Model'] == model) & (results_df['Metric
    ax.errorbar(
        i, row['Mean'], yerr=row['Std'],
        fmt='o', color='black', elinewidth=2, capsize=6
    )

# Add value labels on top of bars
for i, bar in enumerate(ax.patches):
    ax.text(
        bar.get_x() + bar.get_width()/2,
        bar.get_height() + 0.01,
        f"{bar.get_height():.3f}",
        ha='center',
        fontsize=10
    )

plt.title(f'Model Accuracy Comparison with {k_folds}-Fold Cross Validatio
plt.xlabel('Model', fontsize=14)
plt.ylabel('Accuracy', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.ylim([0.5, 1.0])
plt.tight_layout()
plt.show()

# Create a grouped bar chart for all metrics
plt.figure(figsize=(15, 10))
sns.set_style("whitegrid")

# Create grouped bar plot
ax = sns.barplot(
    data=results_df,
```

```python
    x='Model',
    y='Mean',
    hue='Metric',
    order=model_order,
    palette='Set2'
)

plt.title(f'Model Performance Metrics Comparison ({k_folds}-Fold CV)', fo
plt.xlabel('Model', fontsize=14)
plt.ylabel('Score', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.legend(title='Metric', loc='lower right')
plt.ylim([0.5, 1.0])
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
Performing 10-fold cross-validation for Gaussian NB...
  Average: Accuracy=0.6728 ± 0.0245
  Average: F1 Score=0.6090 ± 0.0290

Performing 10-fold cross-validation for Multinomial NB...
  Average: Accuracy=0.7868 ± 0.0232
  Average: F1 Score=0.5587 ± 0.0653

Performing 10-fold cross-validation for Complement NB...
  Average: Accuracy=0.7772 ± 0.0229
  Average: F1 Score=0.6921 ± 0.0204

Performing 10-fold cross-validation for Bernoulli NB...
  Average: Accuracy=0.7982 ± 0.0215
  Average: F1 Score=0.6795 ± 0.0355

Performing 10-fold cross-validation for KNN...
  Average: Accuracy=0.7114 ± 0.0367
  Average: F1 Score=0.5443 ± 0.0647

Performing 10-fold cross-validation for SVM...
  Average: Accuracy=0.7930 ± 0.0319
  Average: F1 Score=0.6562 ± 0.0459

Performing 10-fold cross-validation for Logistic Regression...
  Average: Accuracy=0.7904 ± 0.0262
  Average: F1 Score=0.5900 ± 0.0711

Performing 10-fold cross-validation for Random Forest...
  Average: Accuracy=0.7807 ± 0.0222
  Average: F1 Score=0.5803 ± 0.0515

/tmp/ipykernel_368067/4283035198.py:93: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.

  ax = sns.barplot(
```





Parameter optimization

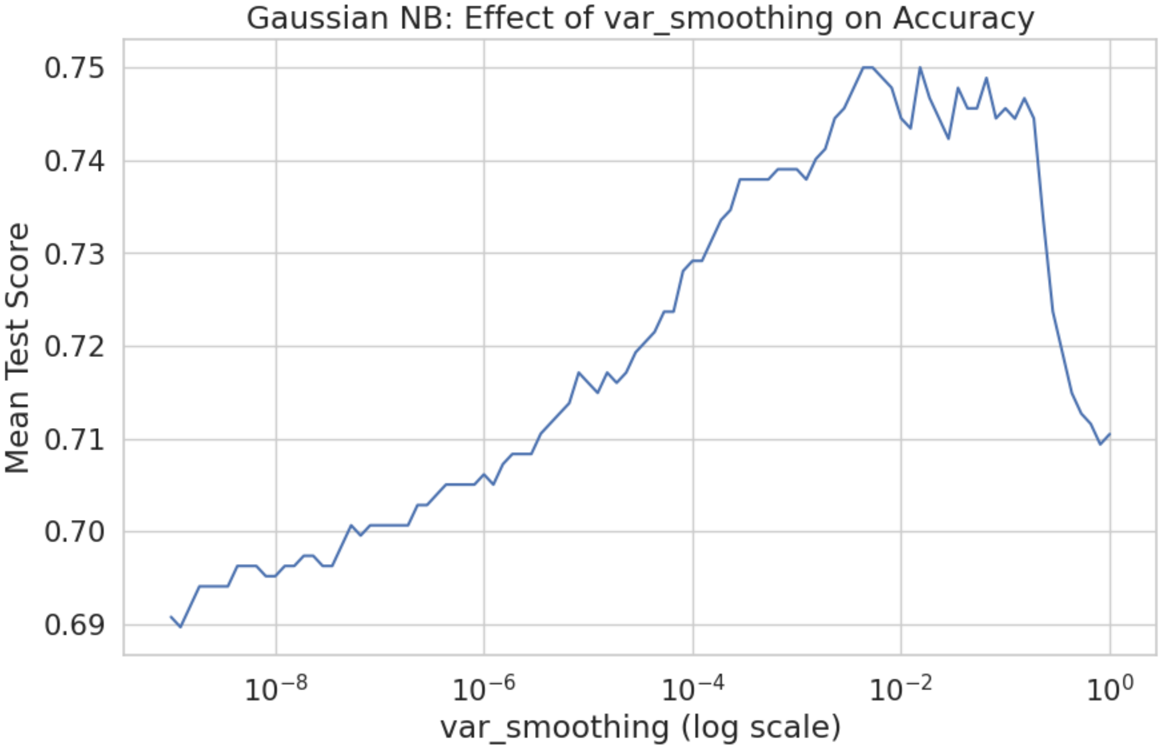
In [90]:
```python
from sklearn.model_selection import GridSearchCV







for model_name, model in nb_models.items():
```

```
    print(f"Performing Grid Search for {model_name}...")
    # Define the parameter grid
    if model_name == 'Gaussian NB':
        param_grid = {
            'var_smoothing': np.
            logspace(0, -9, num=100)
        }
    elif model_name == 'Multinomial NB':
        param_grid = {
            'alpha': np.logspace(-3, 3, num=100),
            'fit_prior': [True, False]
        }
    elif model_name == 'Complement NB':
        param_grid = {
            'alpha': np.logspace(-3, 3, num=100),
            'fit_prior': [True, False]
        }
    elif model_name == 'Bernoulli NB':
        param_grid = {
            'alpha': np.logspace(-3, 3, num=100),
            'fit_prior': [True, False]
        }
    elif model_name == 'KNN':
        param_grid = {
            'n_neighbors': [3, 5, 7, 9],
            'weights': ['uniform', 'distance'],
            'metric': ['euclidean', 'manhattan']
        }
    elif model_name == 'SVM':
        param_grid = {
            'C': np.logspace(-3, 3, num=100),
            'kernel': ['linear', 'rbf'],
            'gamma': ['scale', 'auto']
        }
    elif model_name == 'Logistic Regression':
        param_grid = {
            'C': np.logspace(-3, 3, num=100),
            'penalty': ['l1', 'l2'],
            'solver': ['liblinear', 'saga']
        }
    elif model_name == 'Random Forest':
        param_grid = {
            'n_estimators': [50, 100, 200],
            'max_depth': [None, 10, 20, 30],
            'min_samples_split': [2, 5, 10],
            'min_samples_leaf': [1, 2, 4]
        }
    else:
        continue

    grid_search = GridSearchCV(
        estimator=model,
        param_grid=param_grid,
        scoring='accuracy',
        cv=StratifiedKFold(n_splits=k_folds, shuffle=True, random_state=4
        n_jobs=-1,
        verbose=1
    )
```

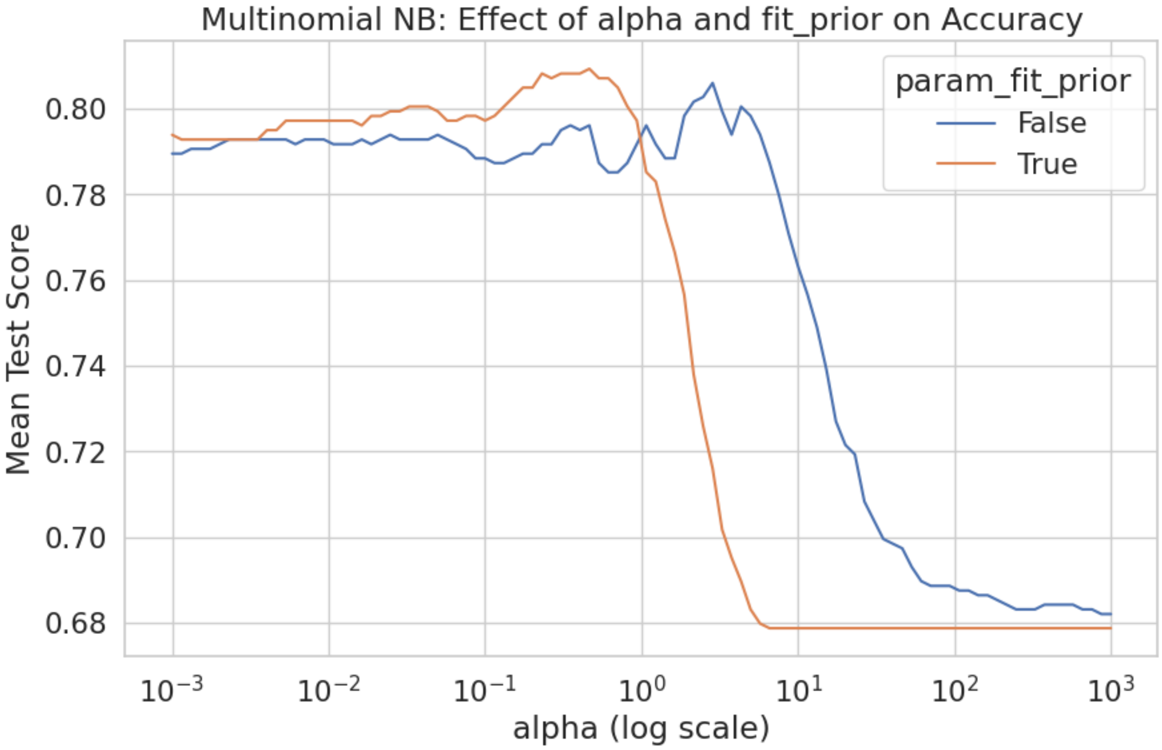
```
# Fit the grid search to the data
grid_search.fit(X_train_task1, y_train_task1)

# Get the best parameters and best score
print(f"Best Parameters: {grid_search.best_params_}")
print(f"Best Cross-Validation Score: {grid_search.best_score_:.4f}")

# Evaluate the best model
best_model = grid_search.best_estimator_
best_model_accuracy = best_model.score(X_test_task1, y_test_task1)
print(f"Test Accuracy with Best Parameters: {best_model_accuracy:.4f}

# Visualize parameter impact
results = pd.DataFrame(grid_search.cv_results_)

# Plot effect of parameters

if model_name == 'Gaussian NB':
    plt.figure(figsize=(10, 6))
    sns.lineplot(data=results, x='param_var_smoothing', y='mean_test_
    plt.xscale('log')
    plt.title(f'Gaussian NB: Effect of var_smoothing on Accuracy')
    plt.xlabel('var_smoothing (log scale)')
    plt.ylabel('Mean Test Score')
    plt.grid(True)
    plt.show()
elif model_name == 'Multinomial NB':
    plt.figure(figsize=(10, 6))
    sns.lineplot(data=results, x='param_alpha', y='mean_test_score',
    plt.xscale('log')
    plt.title(f'Multinomial NB: Effect of alpha and fit_prior on Accu
    plt.xlabel('alpha (log scale)')
    plt.ylabel('Mean Test Score')
    plt.grid(True)
    plt.show()
elif model_name == 'Complement NB':
    plt.figure(figsize=(10, 6))
    sns.lineplot(data=results, x='param_alpha', y='mean_test_score',
    plt.xscale('log')
    plt.title(f'Complement NB: Effect of alpha and fit_prior on Accur
    plt.xlabel('alpha (log scale)')
    plt.ylabel('Mean Test Score')
    plt.grid(True)
    plt.show()
elif model_name == 'Bernoulli NB':
    plt.figure(figsize=(10, 6))
    sns.lineplot(data=results, x='param_alpha', y='mean_test_score',
    plt.xscale('log')
    plt.title(f'Bernoulli NB: Effect of alpha and fit_prior on Accura
    plt.xlabel('alpha (log scale)')
    plt.ylabel('Mean Test Score')
    plt.grid(True)
    plt.show()
elif model_name == 'KNN':
    plt.figure(figsize=(10, 6))
    sns.lineplot(data=results, x='param_n_neighbors', y='mean_test_sc
    plt.title(f'KNN: Effect of n_neighbors and weights on Accuracy')
    plt.xlabel('n_neighbors')
    plt.ylabel('Mean Test Score')
    plt.grid(True)
```

```
        plt.show()
    elif model_name == 'SVM':
        plt.figure(figsize=(10, 6))
        sns.lineplot(data=results, x='param_C', y='mean_test_score', hue=
        plt.xscale('log')
        plt.title(f'SVM: Effect of C and kernel on Accuracy')
        plt.xlabel('C (log scale)')
        plt.ylabel('Mean Test Score')
        plt.grid(True)
        plt.show()
    elif model_name == 'Logistic Regression':
        plt.figure(figsize=(10, 6))
        sns.lineplot(data=results, x='param_C', y='mean_test_score', hue=
        plt.xscale('log')
        plt.title(f'Logistic Regression: Effect of C and penalty on Accur
        plt.xlabel('C (log scale)')
        plt.ylabel('Mean Test Score')
        plt.grid(True)
        plt.show()
    elif model_name == 'Random Forest':
        plt.figure(figsize=(10, 6))
        sns.lineplot(data=results, x='param_n_estimators', y='mean_test_s
        plt.title(f'Random Forest: Effect of n_estimators and max_depth o
        plt.xlabel('n_estimators')
        plt.ylabel('Mean Test Score')
        plt.grid(True)
        plt.show()
    else:
        print(f"No visualization available for {model_name} model.")
```

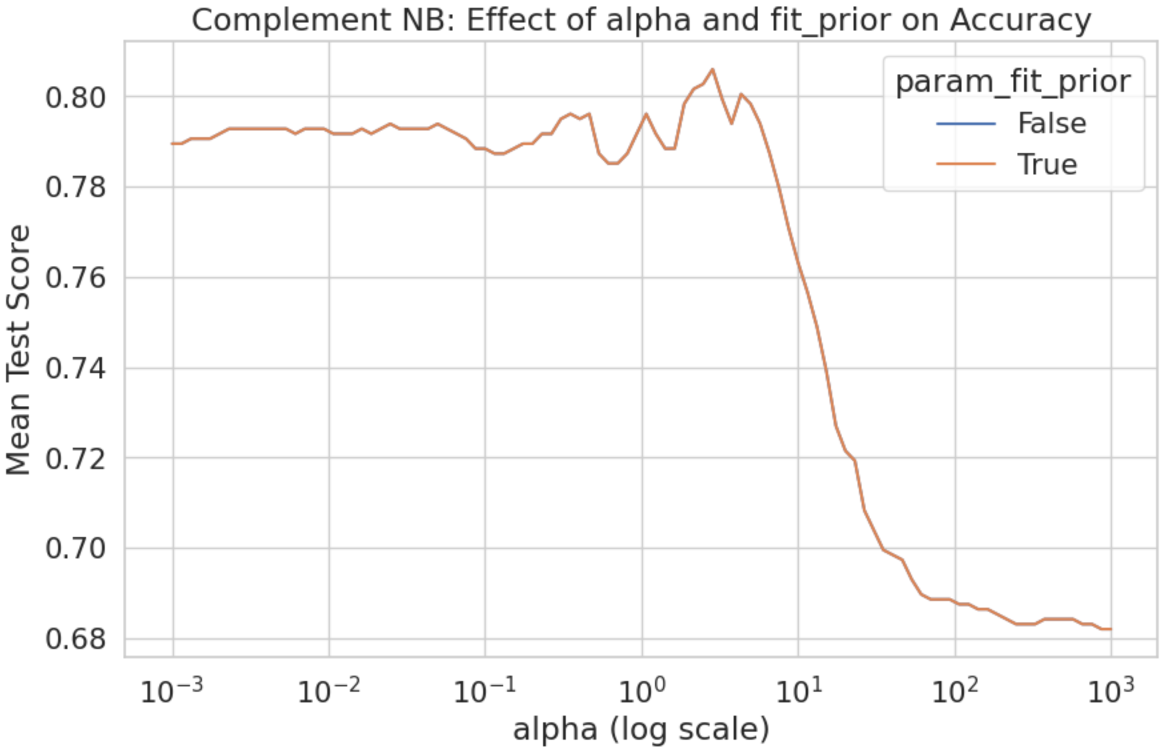
```
Performing Grid Search for Gaussian NB...
Fitting 10 folds for each of 100 candidates, totalling 1000 fits
Best Parameters: {'var_smoothing': np.float64(0.015199110829529332)}
Best Cross-Validation Score: 0.7500
Test Accuracy with Best Parameters: 0.7325
```



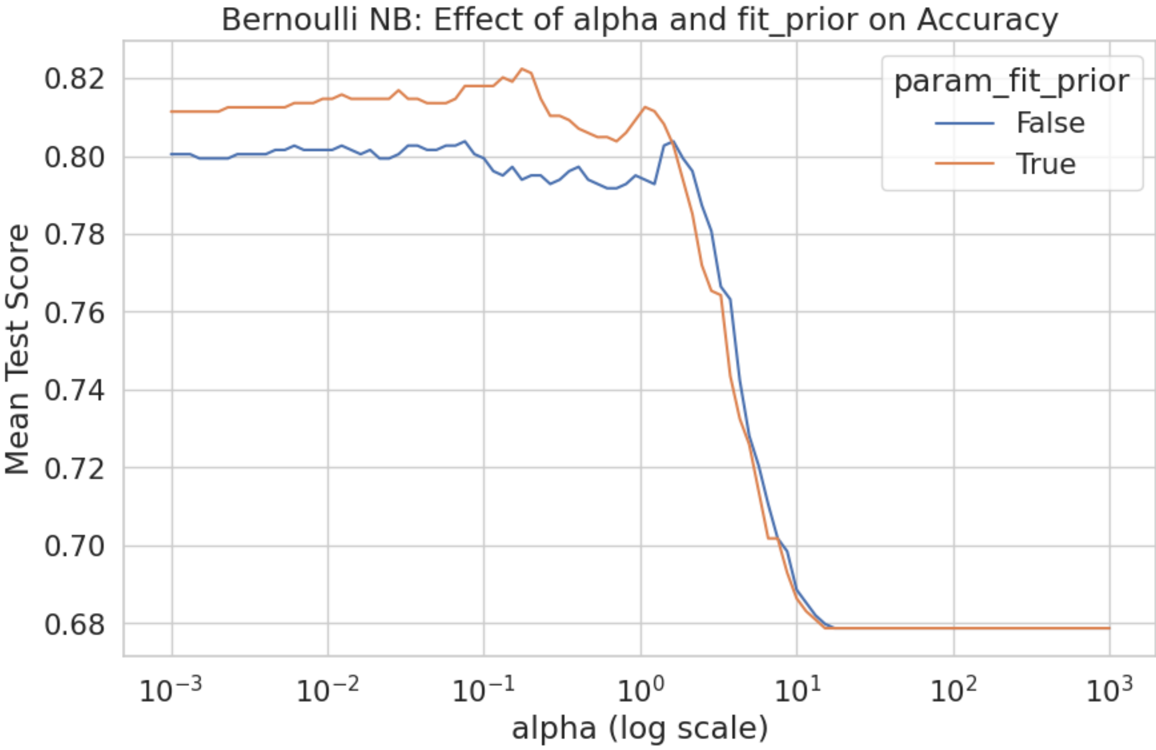
```
Performing Grid Search for Multinomial NB...
Fitting 10 folds for each of 200 candidates, totalling 2000 fits
Best Parameters: {'alpha': np.float64(0.4641588833612782), 'fit_prior': Tr
ue}
Best Cross-Validation Score: 0.8093
Test Accuracy with Best Parameters: 0.7982
```



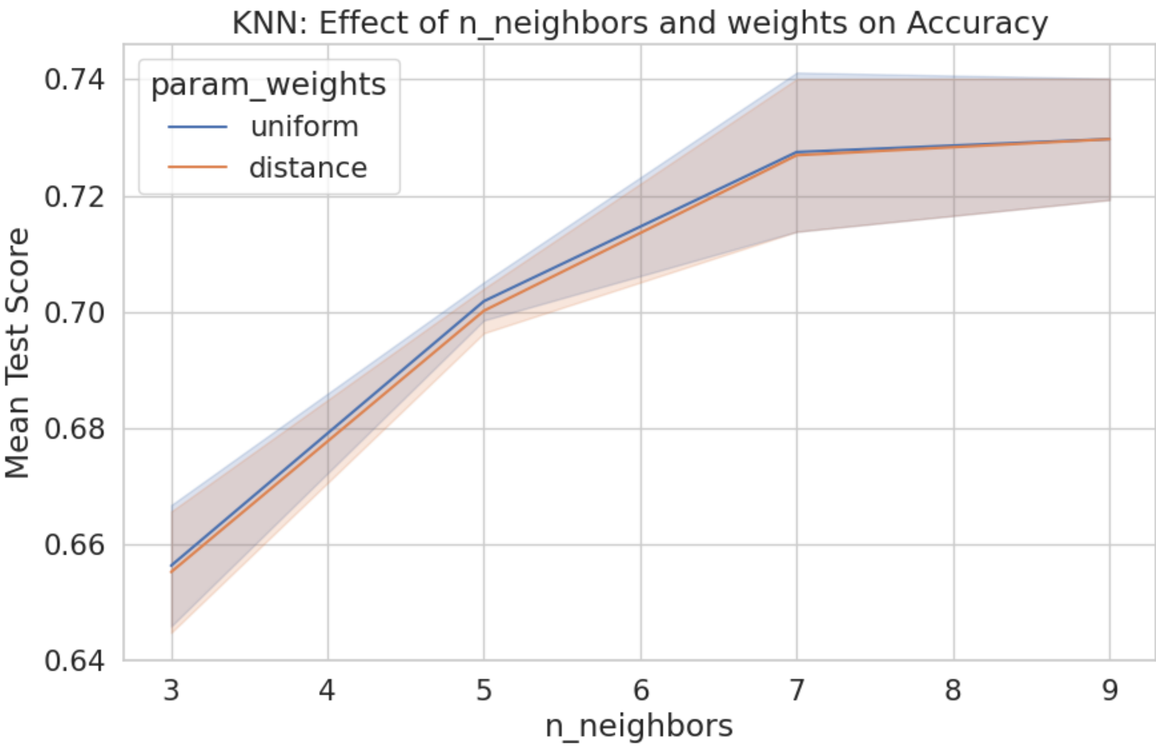

```
Performing Grid Search for Complement NB...
Fitting 10 folds for each of 200 candidates, totalling 2000 fits
Best Parameters: {'alpha': np.float64(2.848035868435802), 'fit_prior': Tru
e}
Best Cross-Validation Score: 0.8060
Test Accuracy with Best Parameters: 0.7895
```



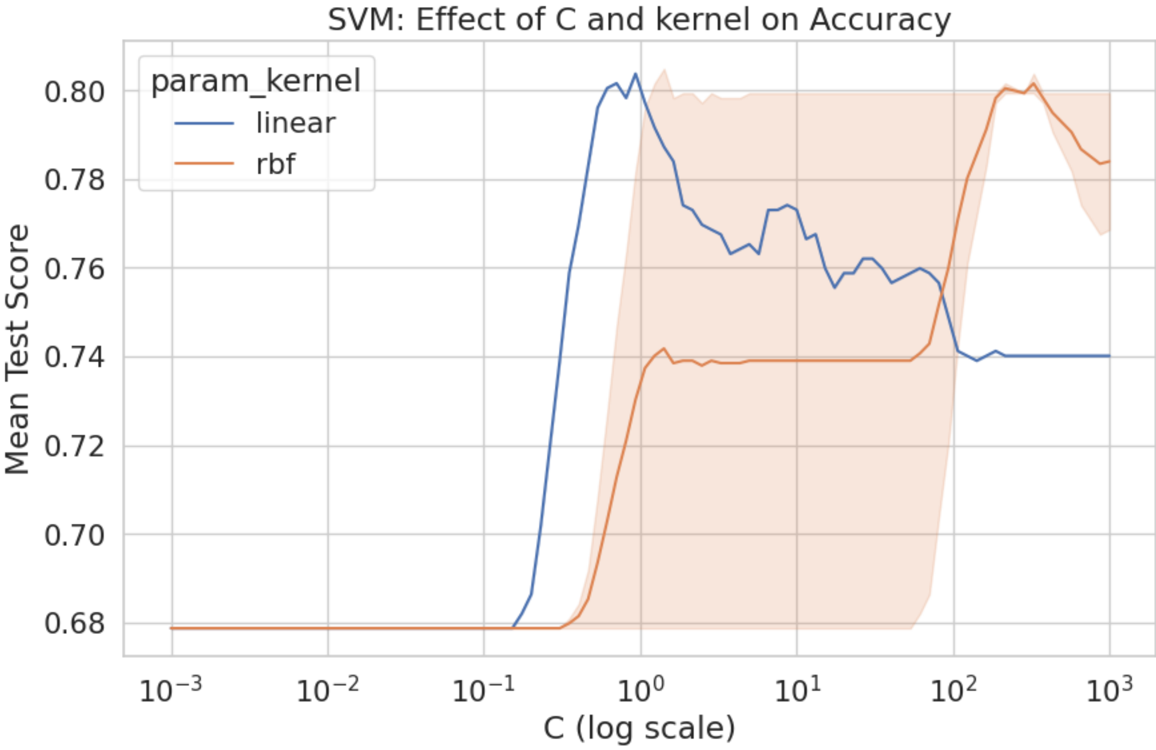
```
Performing Grid Search for Bernoulli NB...
Fitting 10 folds for each of 200 candidates, totalling 2000 fits
Best Parameters: {'alpha': np.float64(0.1747528400007685), 'fit_prior': Tr
ue}
Best Cross-Validation Score: 0.8224
Test Accuracy with Best Parameters: 0.8070
```




```
Performing Grid Search for KNN...
Fitting 10 folds for each of 16 candidates, totalling 160 fits
Best Parameters: {'metric': 'euclidean', 'n_neighbors': 7, 'weights': 'uni
form'}
Best Cross-Validation Score: 0.7412
Test Accuracy with Best Parameters: 0.7237
```



```
Performing Grid Search for SVM...
Fitting 10 folds for each of 400 candidates, totalling 4000 fits
Best Parameters: {'C': np.float64(1.4174741629268048), 'gamma': 'scale', '
kernel': 'rbf'}
Best Cross-Validation Score: 0.8049
Test Accuracy with Best Parameters: 0.7719
```




```
Performing Grid Search for Logistic Regression...
Fitting 10 folds for each of 400 candidates, totalling 4000 fits
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

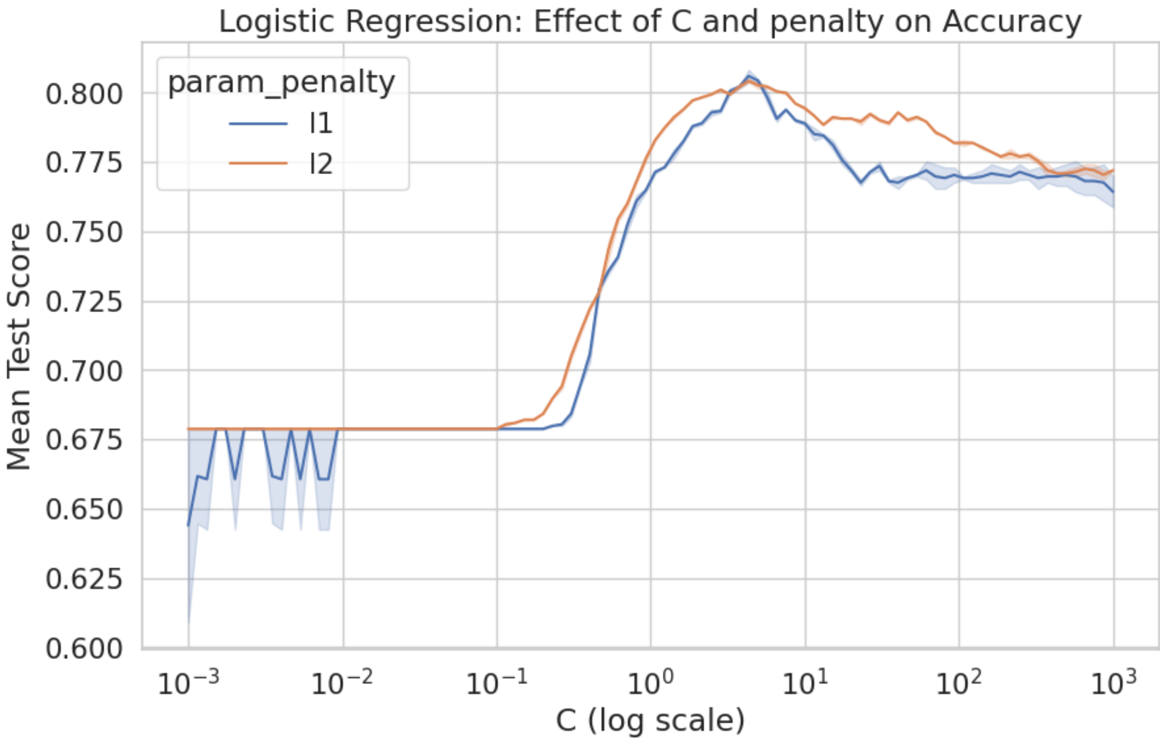
```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

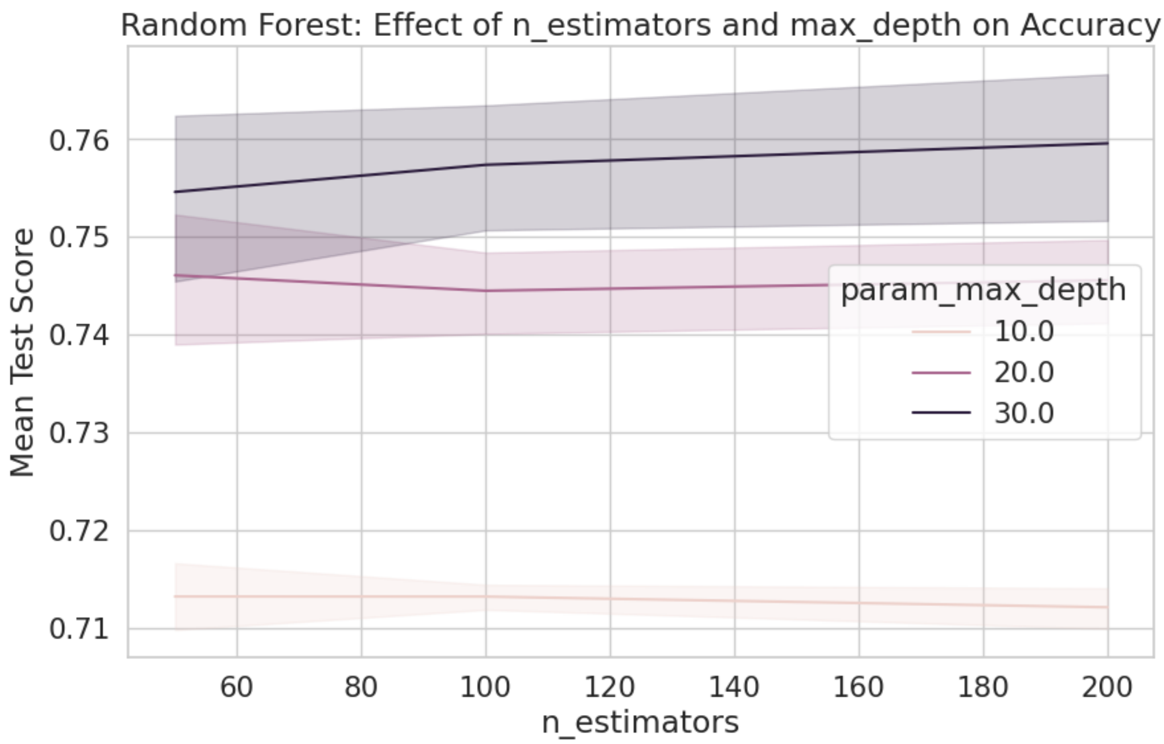
```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reach
ed which means the coef_ did not converge
  warnings.warn(
Best Parameters: {'C': np.float64(4.328761281083062), 'penalty': 'l1', 'so
lver': 'saga'}
Best Cross-Validation Score: 0.8081
Test Accuracy with Best Parameters: 0.7939
```




```
Performing Grid Search for Random Forest...
Fitting 10 folds for each of 108 candidates, totalling 1080 fits
Best Parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_s
plit': 10, 'n_estimators': 50}
Best Cross-Validation Score: 0.7818
Test Accuracy with Best Parameters: 0.7675
```



Training comparaison with best parameter

In [91]:

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Create dictionary of models with their best parameters based on grid se

# Define models with their best parameters based on grid search results
best_models = {
    'Gaussian NB': GaussianNB(var_smoothing=0.0152), #ample value - updat
    'Multinomial NB': MultinomialNB(alpha=0.4641, fit_prior=True),  # Exa
    'Complement NB': ComplementNB(alpha=2.8480, fit_prior=True),  # Examp
    'Bernoulli NB': BernoulliNB(alpha=0.17475, fit_prior=True),  # Exampl
    'KNN': KNeighborsClassifier(n_neighbors=7, weights='uniform', metric=
    'SVM': SVC(C=1.4174742, kernel='rbf', gamma='scale', probability=True
    'Logistic Regression': LogisticRegression(C=np.float64(4.328761281083
    'Random Forest': RandomForestClassifier(n_estimators=200, max_depth=N
}

# Dictionary to store results
best_cv_results = {}

# Perform k-fold cross validation for each model
for model_name, model in best_models.items():
    print(f"Performing {k_folds}-fold cross-validation for {model_name} w

    # Initialize lists to store performance metrics for each fold
    fold_accuracy = []
    fold_precision = []
    fold_recall = []
    fold_f1 = []
    fold_auc = []
```

```
    # For each fold
    for fold, (train_idx, test_idx) in enumerate(skf.split(X_dense, y_tas
        # Split data
        X_train_fold, X_test_fold = X_dense[train_idx], X_dense[test_idx]
        y_train_fold, y_test_fold = y_task1.iloc[train_idx], y_task1.iloc

        # Train model
        model.fit(X_train_fold, y_train_fold)

        # Make predictions
        y_pred_fold = model.predict(X_test_fold)

        # Calculate metrics
        acc = accuracy_score(y_test_fold, y_pred_fold)
        prec = precision_score(y_test_fold, y_pred_fold, zero_division=0)
        rec = recall_score(y_test_fold, y_pred_fold, zero_division=0)
        f1 = f1_score(y_test_fold, y_pred_fold, zero_division=0)

        # For AUC, we need probability estimates
        try:
            if hasattr(model, "predict_proba"):
                y_prob = model.predict_proba(X_test_fold)[:, 1]
                auc_score = roc_auc_score(y_test_fold, y_prob)
                fold_auc.append(auc_score)
        except:
            pass

        fold_accuracy.append(acc)
        fold_precision.append(prec)
        fold_recall.append(rec)
        fold_f1.append(f1)

    # Store average metrics and standard deviations
    best_cv_results[model_name] = {
        'accuracy': {
            'mean': np.mean(fold_accuracy),
            'std': np.std(fold_accuracy)
        },
        'precision': {
            'mean': np.mean(fold_precision),
            'std': np.std(fold_precision)
        },
        'recall': {
            'mean': np.mean(fold_recall),
            'std': np.std(fold_recall)
        },
        'f1': {
            'mean': np.mean(fold_f1),
            'std': np.std(fold_f1)
        }
    }

    if fold_auc:
        best_cv_results[model_name]['auc'] = {
            'mean': np.mean(fold_auc),
            'std': np.std(fold_auc)
        }
        print(f"  Average: AUC={best_cv_results[model_name]['auc']['mean'

    print(f"  Average: Accuracy={best_cv_results[model_name]['accuracy'][
```

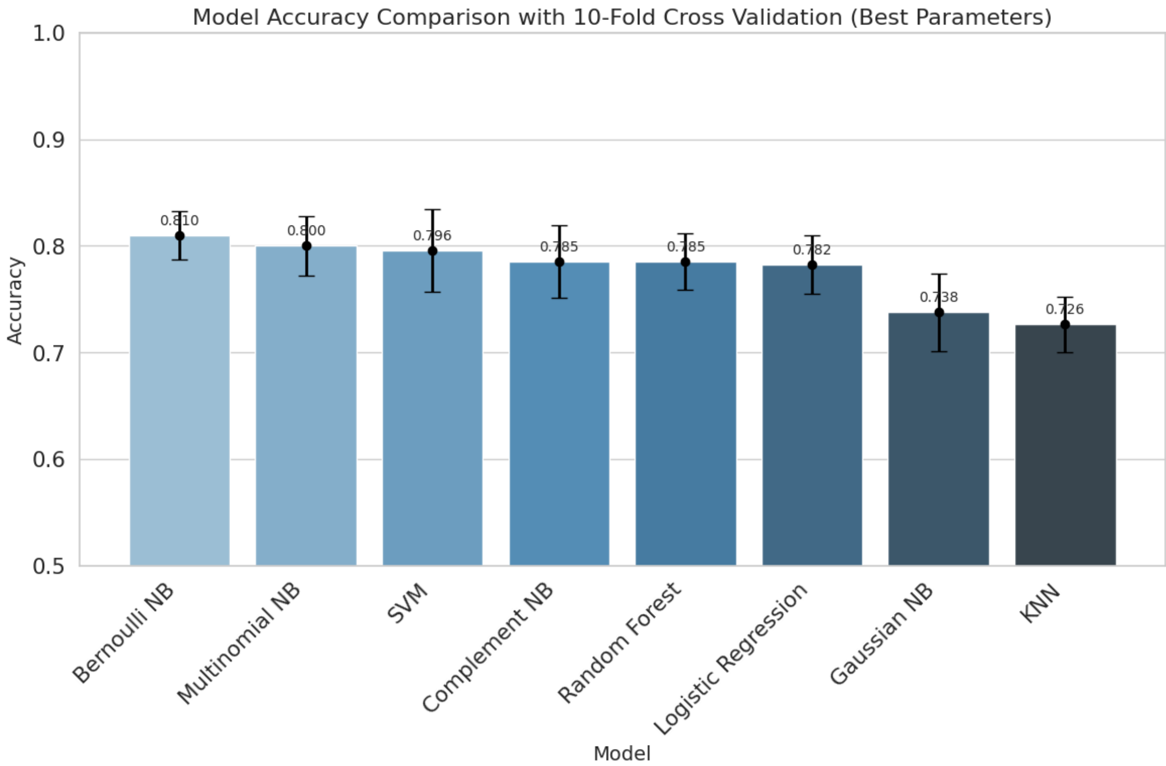
```
    print(f"  Average: F1 Score={best_cv_results[model_name]['f1']['mean'
    print()

# Create DataFrame for visualization
best_results_df = pd.DataFrame({
    'Model': [],
    'Metric': [],
    'Mean': [],
    'Std': []
})

for model_name in best_cv_results:
    for metric in ['accuracy', 'precision', 'recall', 'f1']:
        best_results_df = pd.concat([best_results_df, pd.DataFrame({
            'Model': [model_name],
            'Metric': [metric.capitalize()],
            'Mean': [best_cv_results[model_name][metric]['mean']],
            'Std': [best_cv_results[model_name][metric]['std']]
        })], ignore_index=True)
    if 'auc' in best_cv_results[model_name]:
        best_results_df = pd.concat([best_results_df, pd.DataFrame({
            'Model': [model_name],
            'Metric': ['AUC'],
            'Mean': [best_cv_results[model_name]['auc']['mean']],
            'Std': [best_cv_results[model_name]['auc']['std']]
        })], ignore_index=True)

# Sort models by accuracy
best_model_order = best_results_df[best_results_df['Metric'] == 'Accuracy

# Create plots
plt.figure(figsize=(12, 8))
sns.set_style("whitegrid")

# Create bar plot for accuracy
ax = sns.barplot(
    data=best_results_df[best_results_df['Metric'] == 'Accuracy'],
    x='Model',
    y='Mean',
    order=best_model_order,
    palette='Blues_d'
)

# Add error bars
for i, model in enumerate(best_model_order):
    row = best_results_df[(best_results_df['Model'] == model) & (best_res
    ax.errorbar(
        i, row['Mean'], yerr=row['Std'],
        fmt='o', color='black', elinewidth=2, capsize=6
    )

# Add value labels on top of bars
for i, bar in enumerate(ax.patches):
    ax.text(
        bar.get_x() + bar.get_width()/2,
        bar.get_height() + 0.01,
        f"{bar.get_height():.3f}",
        ha='center',
        fontsize=10
    )
```

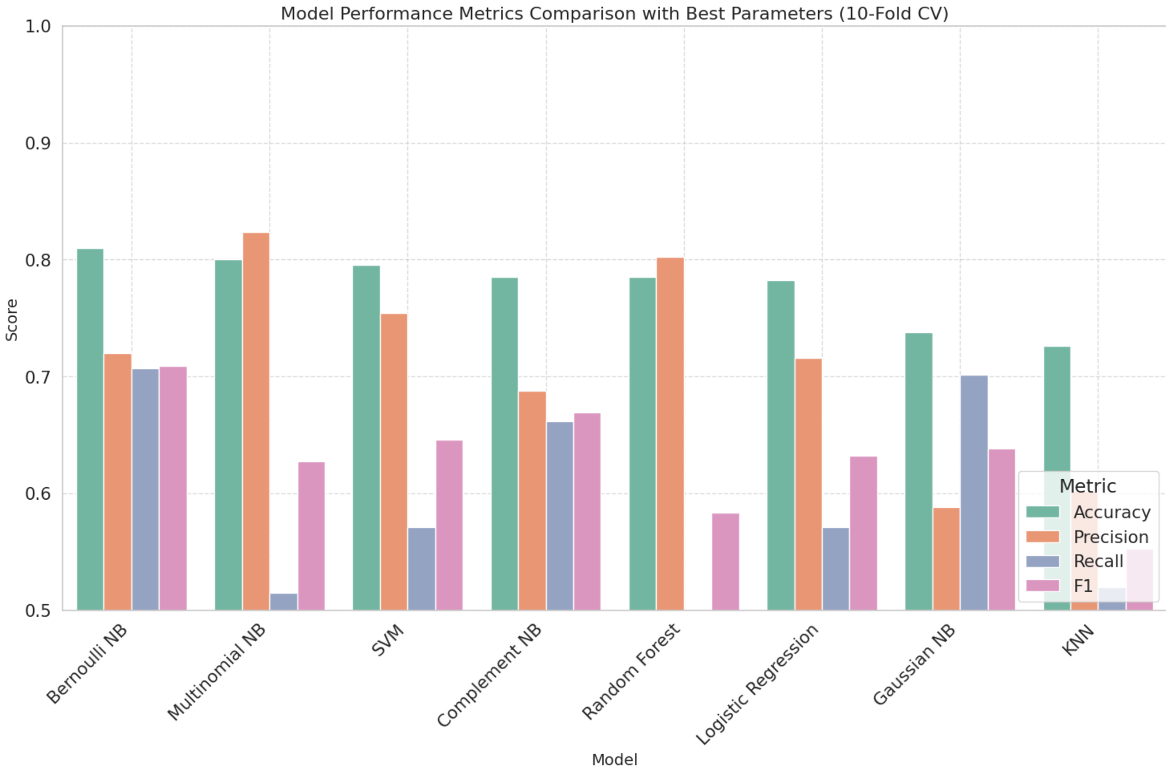
```
plt.title(f'Model Accuracy Comparison with {k_folds}-Fold Cross Validatio
plt.xlabel('Model', fontsize=14)
plt.ylabel('Accuracy', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.ylim([0.5, 1.0])
plt.tight_layout()
plt.show()

# Create a grouped bar chart for all metrics
plt.figure(figsize=(15, 10))
sns.set_style("whitegrid")

# Create grouped bar plot
ax = sns.barplot(
    data=best_results_df[best_results_df['Metric'] != 'AUC'],
    x='Model',
    y='Mean',
    hue='Metric',
    order=best_model_order,
    palette='Set2'
)

plt.title(f'Model Performance Metrics Comparison with Best Parameters ({k
plt.xlabel('Model', fontsize=14)
plt.ylabel('Score', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.legend(title='Metric', loc='lower right')
plt.ylim([0.5, 1.0])
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Create a comparison table between original results and optimized result
comparison_df = pd.DataFrame({
    'Model': [],
    'Metric': [],
    'Original Mean': [],
    'Original Std': [],
    'Optimized Mean': [],
    'Optimized Std': [],
    'Improvement': []
})

for model_name in best_cv_results:
    if model_name in cv_results:
        for metric in ['accuracy', 'precision', 'recall', 'f1']:
            orig_mean = cv_results[model_name][metric]['mean']
            orig_std = cv_results[model_name][metric]['std']
            opt_mean = best_cv_results[model_name][metric]['mean']
            opt_std = best_cv_results[model_name][metric]['std']
            improvement = ((opt_mean - orig_mean) / orig_mean) * 100

            comparison_df = pd.concat([comparison_df, pd.DataFrame({
                'Model': [model_name],
                'Metric': [metric.capitalize()],
                'Original Mean': [orig_mean],
                'Original Std': [orig_std],
                'Optimized Mean': [opt_mean],
                'Optimized Std': [opt_std],
```

```
                'Improvement': [improvement]
            })], ignore_index=True)

# Plot the improvement in accuracy
plt.figure(figsize=(15, 10))
sns.set_style("whitegrid")

# Filter for accuracy only
acc_comparison = comparison_df[comparison_df['Metric'] == 'Accuracy']
acc_comparison = acc_comparison.sort_values('Improvement', ascending=Fals

# Create bar chart of improvements
ax = sns.barplot(
    data=acc_comparison,
    x='Model',
    y='Improvement',
    palette='RdYlGn',
    dodge=False
)

plt.axhline(y=0, color='black', linestyle='-', alpha=0.3)

# Add value labels on bars
for i, bar in enumerate(ax.patches):
    if bar.get_height() >= 0:
        ax.text(
            bar.get_x() + bar.get_width()/2,
            bar.get_height() + 0.5,
            f"{bar.get_height():.2f}%",
            ha='center',
            fontsize=10
        )
    else:
        ax.text(
            bar.get_x() + bar.get_width()/2,
            bar.get_height() - 1.0,
            f"{bar.get_height():.2f}%",
            ha='center',
            fontsize=10
        )

plt.title('Percentage Improvement in Accuracy After Parameter Optimizatio
plt.xlabel('Model', fontsize=14)
plt.ylabel('Improvement (%)', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

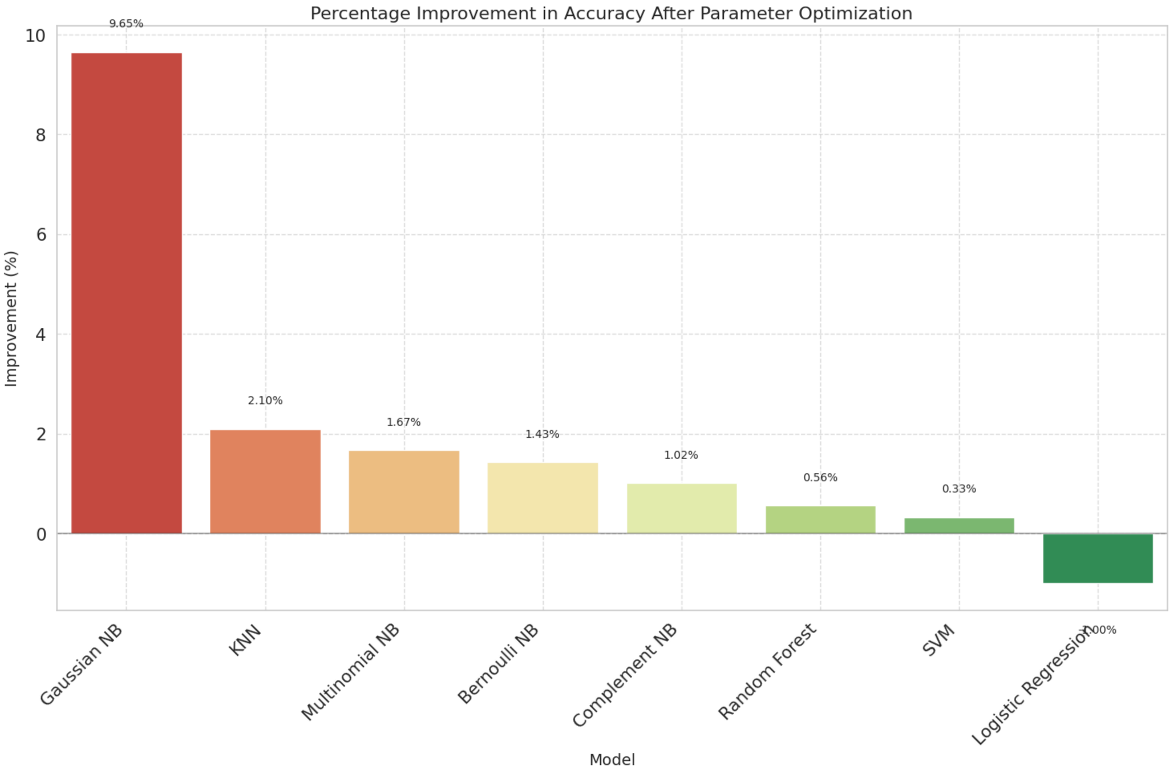
```
Performing 10-fold cross-validation for Gaussian NB with best parameter
s...
  Average: AUC=0.7703 ± 0.0396
  Average: Accuracy=0.7377 ± 0.0363
  Average: F1 Score=0.6381 ± 0.0403

Performing 10-fold cross-validation for Multinomial NB with best parameter
s...
  Average: AUC=0.8660 ± 0.0281
  Average: Accuracy=0.8000 ± 0.0277
  Average: F1 Score=0.6273 ± 0.0495

Performing 10-fold cross-validation for Complement NB with best parameter
s...
  Average: AUC=0.8549 ± 0.0274
  Average: Accuracy=0.7851 ± 0.0343
  Average: F1 Score=0.6695 ± 0.0449

Performing 10-fold cross-validation for Bernoulli NB with best parameter
s...
  Average: AUC=0.8681 ± 0.0237
  Average: Accuracy=0.8096 ± 0.0229
  Average: F1 Score=0.7092 ± 0.0316

Performing 10-fold cross-validation for KNN with best parameters...
  Average: AUC=0.7673 ± 0.0471
  Average: Accuracy=0.7263 ± 0.0260
  Average: F1 Score=0.5524 ± 0.0539

Performing 10-fold cross-validation for SVM with best parameters...
  Average: AUC=0.8655 ± 0.0213
  Average: Accuracy=0.7956 ± 0.0390
  Average: F1 Score=0.6462 ± 0.0696

Performing 10-fold cross-validation for Logistic Regression with best para
meters...
  Average: AUC=0.8541 ± 0.0249
  Average: Accuracy=0.7825 ± 0.0277
  Average: F1 Score=0.6320 ± 0.0482

Performing 10-fold cross-validation for Random Forest with best parameter
s...
  Average: AUC=0.8466 ± 0.0334
  Average: Accuracy=0.7851 ± 0.0264
  Average: F1 Score=0.5831 ± 0.0627
```

```
/tmp/ipykernel_368067/2330540877.py:127: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.

  ax = sns.barplot(
```





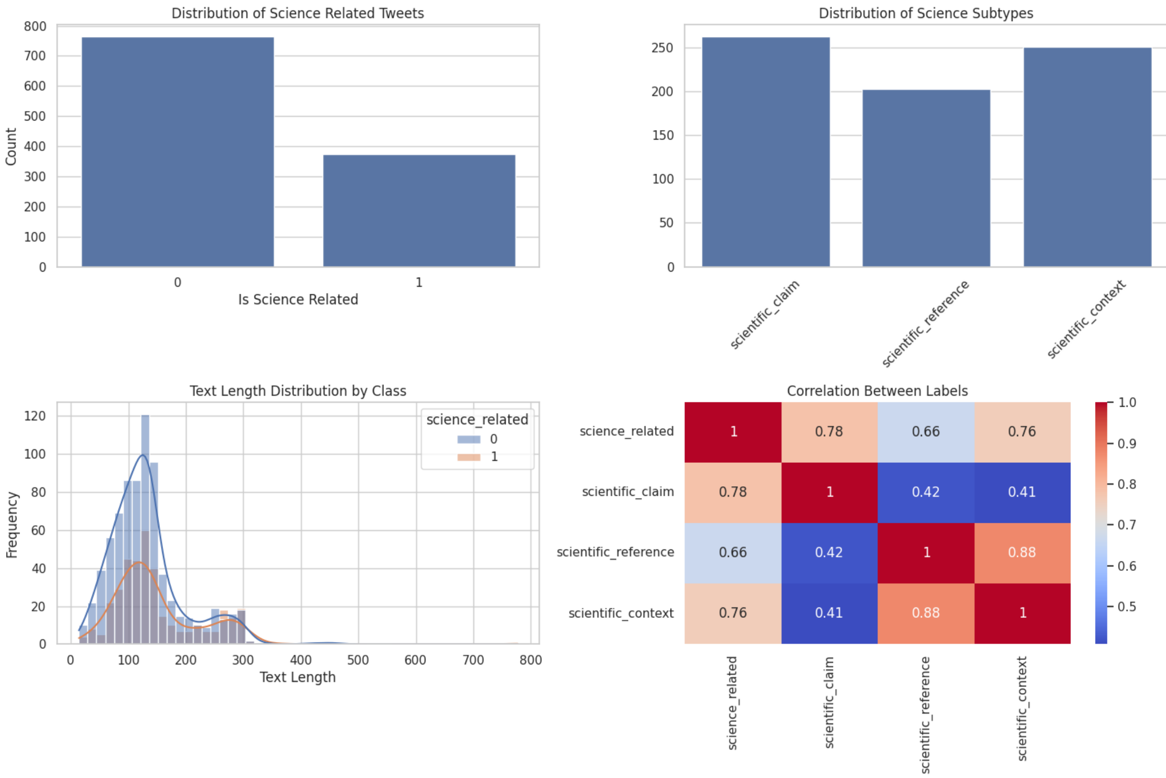

```
/tmp/ipykernel_368067/2330540877.py:224: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.

  ax = sns.barplot(
```



Preprocessing

In [92]:
```
import numpy as np

# Set style
sns.set(style="whitegrid")
plt.figure(figsize=(15, 10))

# Plot distribution of science_related tweets
plt.subplot(2, 2, 1)
sns.countplot(x='science_related', data=df)
plt.title('Distribution of Science Related Tweets')
plt.xlabel('Is Science Related')
plt.ylabel('Count')

# Plot distribution of science subtypes for science-related tweets
sci_df = df[df['science_related'] == 1]
plt.subplot(2, 2, 2)
subtypes = ['scientific_claim', 'scientific_reference', 'scientific_conte
sns.barplot(x=subtypes, y=[sci_df[col].sum() for col in subtypes])
plt.title('Distribution of Science Subtypes')
plt.xticks(rotation=45)
plt.tight_layout()

# Plot text length distribution
plt.subplot(2, 2, 3)
df['text_length'] = df['text'].apply(len)
sns.histplot(data=df, x='text_length', hue='science_related', bins=50, kd
plt.title('Text Length Distribution by Class')
plt.xlabel('Text Length')
plt.ylabel('Frequency')

# Plot correlation between features
plt.subplot(2, 2, 4)
```

```
corr_cols = ['science_related', 'scientific_claim', 'scientific_reference
sns.heatmap(df[corr_cols].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Between Labels')

plt.tight_layout()
plt.show()
```



In [93]:

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
import emoji

# Download required NLTK resources
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('punkt_tab')

def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()

    # Demojize text
    text = emoji.demojize(text)

    # Remove URLs
    text = re.sub(r'(http\S+|www\S+)', '', text)



    # Remove mentions and hashtags exececpt for the word uurekamag
    text = re.sub(r'@\w+', '', text)
    text = re.sub(r'#eurekamag', 'eurekamag', text)
    text = re.sub(r'#\w+', '', text)
```

```
    # Remove special characters and numbers
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenize
    tokens = nltk.word_tokenize(text)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]


    # Lemmatize
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    # Rejoin
    return ' '.join(tokens)

# Apply preprocessing to the dataset
df['processed_text'] = df['text'].apply(preprocess_text)

# Compare before and after
comparison_df = pd.DataFrame({
    'Original': df['text'].head(5),
    'Preprocessed': df['processed_text'].head(5)
})
comparison_df
```

```
[nltk_data] Downloading package stopwords to /home/hurel/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /home/hurel/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /home/hurel/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /home/hurel/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

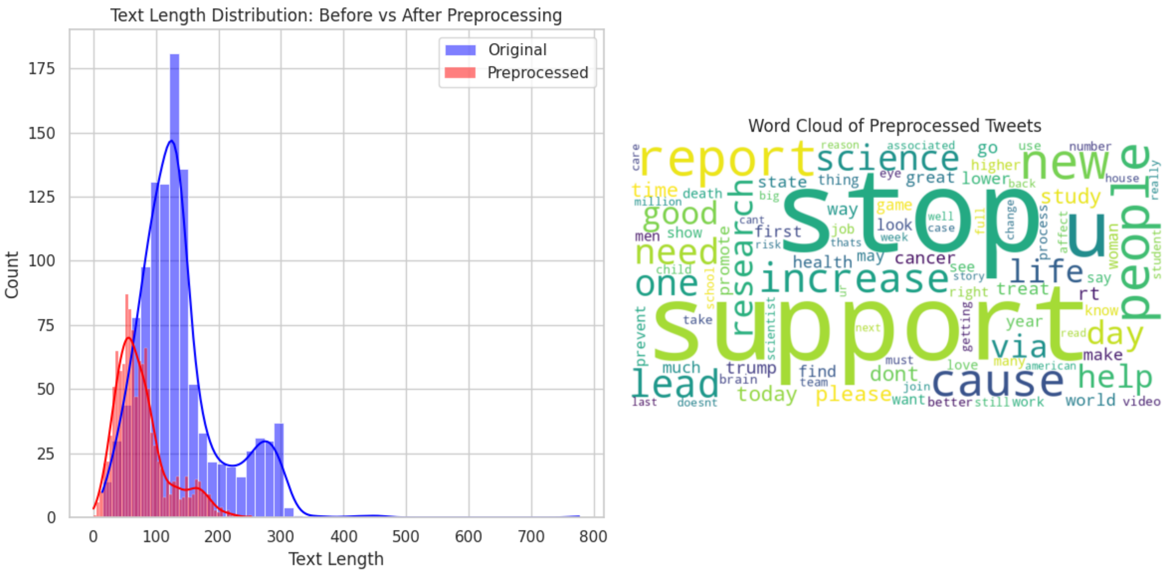
Out[93]:

	Original	Preprocessed
0	Knees are a bit sore. i guess that's a sign th...	knee bit sore guess thats sign recent treadmil...
1	McDonald's breakfast stop then the gym 🏀💪	mcdonalds breakfast stop gym basketballflexedb...
2	Can any Gynecologist with Cancer Experience ex...	gynecologist cancer experience explain danger ...
3	Couch-lock highs lead to sleeping in the couch...	couchlock high lead sleeping couch got ta stop...
4	Does daily routine help prevent problems with ...	daily routine help prevent problem bipolar dis...

In [94]:

```
# Visualize the impact of preprocessing
plt.figure(figsize=(12, 6))

# Text length before and after preprocessing
df['original_length'] = df['text'].apply(len)
df['processed_length'] = df['processed_text'].apply(len)

plt.subplot(1, 2, 1)
```

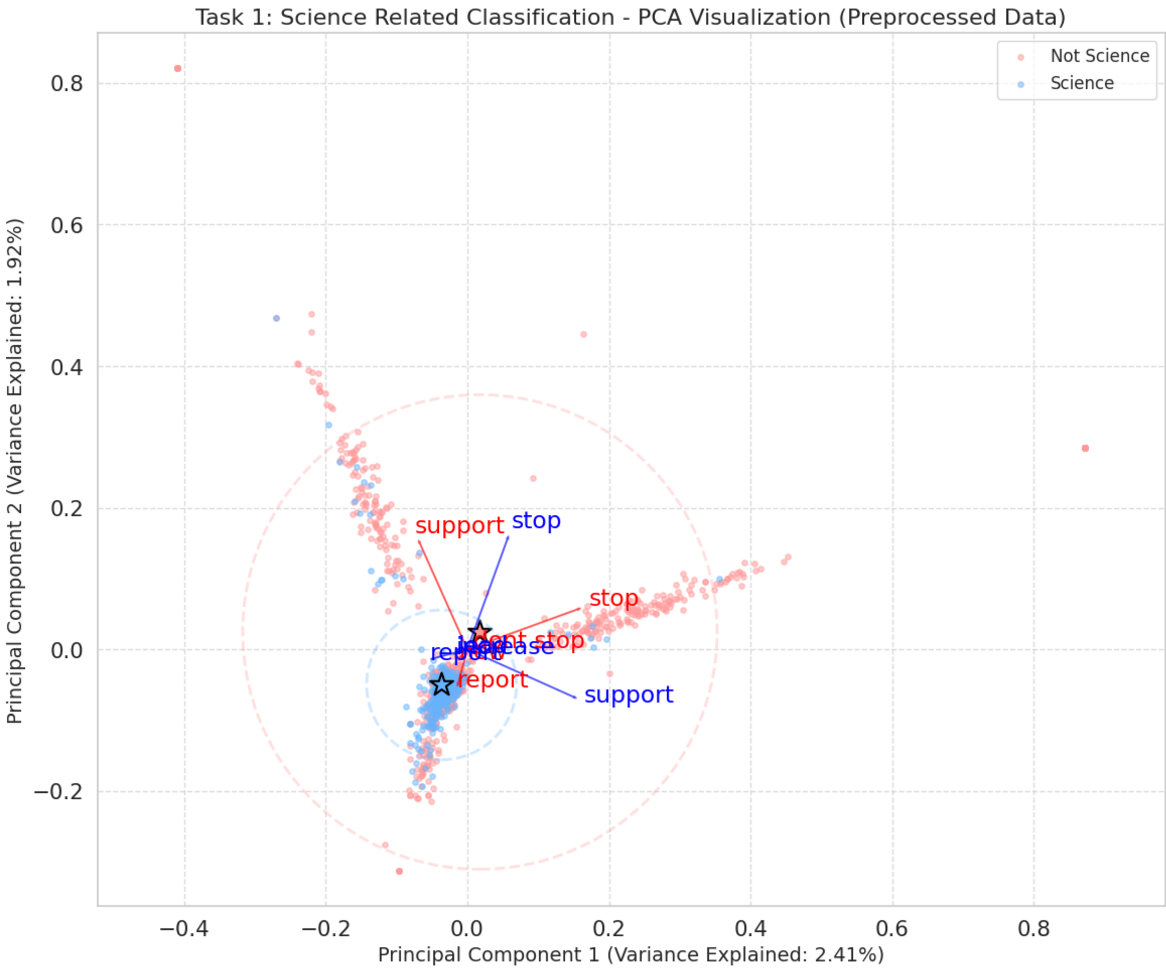
```
sns.histplot(data=df, x='original_length', color='blue', bins=50, kde=Tru
sns.histplot(data=df, x='processed_length', color='red', bins=50, kde=Tru
plt.title('Text Length Distribution: Before vs After Preprocessing')
plt.xlabel('Text Length')
plt.legend()

# Word cloud for processed text
from wordcloud import WordCloud

plt.subplot(1, 2, 2)
wordcloud = WordCloud(width=800, height=400, background_color='white', ma
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Preprocessed Tweets')

plt.tight_layout()
plt.show()
```



In [143…

```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np
from sklearn.decomposition import PCA

# ================= TASK 1 VISUALIZATION WITH POST-PROCESSED DATA =======
# Get the processed data features
X_dense_processed = X_tfidf.toarray()  # Using TF-IDF with preprocessed t

# Get the labels for Task 1
y_task1 = df['science_related']

# Apply PCA
pca_task1 = PCA(n_components=2)
X_2d_task1 = pca_task1.fit_transform(X_dense_processed)

# Create a figure
plt.figure(figsize=(12, 10))

# Define colors for each class in Task 1
colors_task1 = ['#ff9999', '#66b3ff']  # Red, Blue
labels_unique_task1 = [0, 1]
label_names_task1 = ['Not Science', 'Science']
```

```
# Plot points by class
for i, label in enumerate(labels_unique_task1):
    # Get indices for this class
    indices = y_task1 == label

    # Plot points for this class
    plt.scatter(X_2d_task1[indices, 0], X_2d_task1[indices, 1],
                c=colors_task1[i], alpha=0.5, s=15,
                label=label_names_task1[i])

    # Calculate and plot centroid
    centroid = X_2d_task1[indices].mean(axis=0)
    plt.scatter(centroid[0], centroid[1],
                marker='*', s=300, c=colors_task1[i],
                edgecolor='black', linewidth=1.5)

    # Draw a circle around majority of points in this class
    std_dev = X_2d_task1[indices].std(axis=0).mean() * 2  # 2 std dev cir
    circle = plt.Circle((centroid[0], centroid[1]), std_dev,
                        color=colors_task1[i], fill=False,
                        linestyle='--', linewidth=2, alpha=0.3)
    plt.gca().add_patch(circle)

# Add vector directions of top features
feature_names = tfidf_vec.get_feature_names_out()
pca_components = pca_task1.components_

# Get top influential features in each principal component
n_top_features = 10
sorted_idx = np.argsort(-np.abs(pca_components), axis=1)[:, :n_top_featur

# Scale for the arrows
scale = np.abs(X_2d_task1).max() * 0.2

# Plot feature vectors
for i, (idx, c) in enumerate(zip(sorted_idx, ['red', 'blue'])):
    for j, feature_idx in enumerate(idx):
        feature_name = feature_names[feature_idx]
        # Only plot first few to avoid clutter
        if j < 5:
            plt.arrow(0, 0,
                      pca_components[i, feature_idx] * scale,
                      pca_components[(i+1)%2, feature_idx] * scale,
                      color=c, alpha=0.5)
            plt.text(pca_components[i, feature_idx] * scale * 1.1,
                     pca_components[(i+1)%2, feature_idx] * scale * 1.1,
                     feature_name, color=c)

plt.title('Task 1: Science Related Classification - PCA Visualization (Pr
plt.xlabel(f'Principal Component 1 (Variance Explained: {pca_task1.explai
plt.ylabel(f'Principal Component 2 (Variance Explained: {pca_task1.explai
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=12)
plt.axis('equal')  # Equal scaling for x and y
plt.tight_layout()
plt.show()
```

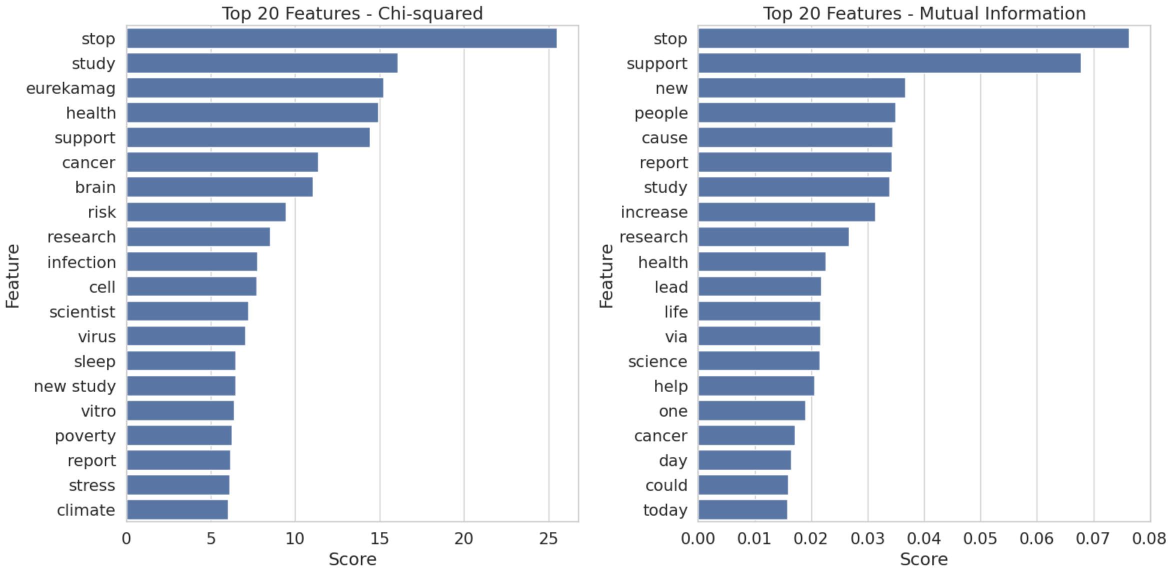


In [95]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X_all = vectorizer.fit_transform(df['text'])
```

In [121…
```
# Compare different feature extraction methods
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_selection import SelectKBest, chi2, mutual_info_clas

# Create label for Task 1
y_task1 = df['science_related']

# 1. Count Vectorizer
count_vec = CountVectorizer(max_features=5000)
X_count = count_vec.fit_transform(df['processed_text'])

# 2. TF-IDF with more parameters
tfidf_vec = TfidfVectorizer(max_features=5000,
                           min_df=5,
                           max_df=0.8,
                           ngram_range=(1, 2))
X_tfidf = tfidf_vec.fit_transform(df['processed_text'])

# 3. TF-IDF with preprocessing already done
tfidf_processed = TfidfVectorizer(max_features=5000)
X_tfidf_processed = tfidf_processed.fit_transform(df['processed_text'])

# Compare feature extraction methods
print(f"Count Vectorizer Features: {X_count.shape}")
print(f"TF-IDF Vectorizer Features: {X_tfidf.shape}")
```

```
print(f"TF-IDF on Preprocessed Text Features: {X_tfidf_processed.shape}")

# Feature selection using Chi-squared
selector_chi2 = SelectKBest(chi2, k=100)
X_chi2 = selector_chi2.fit_transform(X_tfidf, y_task1)

# Feature selection using Mutual Information
selector_mi = SelectKBest(mutual_info_classif, k=100)
X_mi = selector_mi.fit_transform(X_tfidf, y_task1)

print(f"\nFeatures after Chi-squared selection: {X_chi2.shape}")
print(f"Features after Mutual Information selection: {X_mi.shape}")

# Get and visualize the most important features
chi2_selected_indices = selector_chi2.get_support(indices=True)
mi_selected_indices = selector_mi.get_support(indices=True)

chi2_feature_names = np.array(tfidf_vec.get_feature_names_out())[chi2_sel
mi_feature_names = np.array(tfidf_vec.get_feature_names_out())[mi_selecte

# Plot top 20 features by importance
plt.figure(figsize=(16, 8))

plt.subplot(1, 2, 1)
chi2_scores = selector_chi2.scores_[chi2_selected_indices]
chi2_features_df = pd.DataFrame({'Feature': chi2_feature_names, 'Score':
chi2_features_df = chi2_features_df.sort_values('Score', ascending=False)
sns.barplot(x='Score', y='Feature', data=chi2_features_df)
plt.title('Top 20 Features - Chi-squared')

plt.subplot(1, 2, 2)
mi_scores = selector_mi.scores_[mi_selected_indices]
mi_features_df = pd.DataFrame({'Feature': mi_feature_names, 'Score': mi_s
mi_features_df = mi_features_df.sort_values('Score', ascending=False).hea
sns.barplot(x='Score', y='Feature', data=mi_features_df)
plt.title('Top 20 Features - Mutual Information')

plt.tight_layout()
plt.show()

# We'll use the TF-IDF on preprocessed text for subsequent modeling
X_selected = X_tfidf
```

```
Count Vectorizer Features: (1140, 5000)
TF-IDF Vectorizer Features: (1140, 465)
TF-IDF on Preprocessed Text Features: (1140, 5000)

Features after Chi-squared selection: (1140, 100)
Features after Mutual Information selection: (1140, 100)
```



In [97]:

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, MultinomialNB, ComplementNB,
from sklearn.metrics import classification_report, confusion_matrix, accu

# Function to evaluate model performance
def evaluate_model(model, X_train, X_test, y_train, y_test, model_name):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True)
    cm = confusion_matrix(y_test, y_pred)

    print(f"Model: {model_name}")
    print(f"Accuracy: {accuracy:.4f}")
    print(classification_report(y_test, y_pred))

    return {
        'model_name': model_name,
        'accuracy': accuracy,
        'report': report,
        'confusion_matrix': cm,
        'y_pred': y_pred
    }

# Get a dense version of our features for Gaussian NB
X_dense = X_selected.toarray()

# Split data for Task 1
X_train_task1, X_test_task1, y_train_task1, y_test_task1 = train_test_spl
    X_dense, df['task1_label'], test_size=0.2, random_state=42
)

# Define models with their best parameters based on grid search results
nb_models = {
    'Gaussian NB': GaussianNB(var_smoothing=0.0152), #ample value - updat
    'Multinomial NB': MultinomialNB(alpha=0.4641, fit_prior=True),  # Exa
    'Complement NB': ComplementNB(alpha=2.8480, fit_prior=True),  # Examp
    'Bernoulli NB': BernoulliNB(alpha=0.17475, fit_prior=True),  # Exampl
    'KNN': KNeighborsClassifier(n_neighbors=7, weights='uniform', metric=
    'SVM': SVC(C=1.4174742, kernel='rbf', gamma='scale', probability=True
    'Logistic Regression': LogisticRegression(C=np.float64(4.328761281083
```

```
    'Random Forest': RandomForestClassifier(n_estimators=200, max_depth=N
}

task1_results = {}
print("Task 1: Science Related Classification\n" + "="*40)
for name, model in nb_models.items():
    task1_results[name] = evaluate_model(
        model, X_train_task1, X_test_task1, y_train_task1, y_test_task1,
    )
    print("\n")
```

```
Task 1: Science Related Classification
========================================
Model: Gaussian NB
Accuracy: 0.6798
              precision    recall  f1-score   support

           0       0.80      0.66      0.73       146
           1       0.54      0.71      0.61        82

    accuracy                           0.68       228
   macro avg       0.67      0.69      0.67       228
weighted avg       0.71      0.68      0.69       228



Model: Multinomial NB
Accuracy: 0.7412
              precision    recall  f1-score   support

           0       0.75      0.89      0.82       146
           1       0.71      0.48      0.57        82

    accuracy                           0.74       228
   macro avg       0.73      0.68      0.69       228
weighted avg       0.74      0.74      0.73       228



Model: Complement NB
Accuracy: 0.7500
              precision    recall  f1-score   support

           0       0.82      0.78      0.80       146
           1       0.64      0.70      0.67        82

    accuracy                           0.75       228
   macro avg       0.73      0.74      0.73       228
weighted avg       0.76      0.75      0.75       228



Model: Bernoulli NB
Accuracy: 0.7500
              precision    recall  f1-score   support

           0       0.78      0.86      0.81       146
           1       0.69      0.56      0.62        82

    accuracy                           0.75       228
   macro avg       0.73      0.71      0.72       228
weighted avg       0.74      0.75      0.74       228



Model: KNN
Accuracy: 0.4737
              precision    recall  f1-score   support

           0       0.91      0.20      0.33       146
           1       0.40      0.96      0.57        82
```

```
    accuracy                           0.47       228
   macro avg       0.65      0.58      0.45       228
weighted avg       0.73      0.47      0.41       228



Model: SVM
Accuracy: 0.7763
              precision    recall  f1-score   support

           0       0.78      0.90      0.84       146
           1       0.75      0.56      0.64        82

    accuracy                           0.78       228
   macro avg       0.77      0.73      0.74       228
weighted avg       0.77      0.78      0.77       228



Model: Logistic Regression
Accuracy: 0.7675
              precision    recall  f1-score   support

           0       0.79      0.88      0.83       146
           1       0.72      0.57      0.64        82

    accuracy                           0.77       228
   macro avg       0.75      0.72      0.73       228
weighted avg       0.76      0.77      0.76       228



Model: Random Forest
Accuracy: 0.7588
              precision    recall  f1-score   support

           0       0.83      0.79      0.81       146
           1       0.65      0.71      0.68        82

    accuracy                           0.76       228
   macro avg       0.74      0.75      0.74       228
weighted avg       0.76      0.76      0.76       228
```

In [98]:
```
# Using preprocessed text for feature extraction
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Create TF-IDF features from preprocessed text
tfidf_vec = TfidfVectorizer(max_features=5000,
                           min_df=5,
                           max_df=0.8,
                           ngram_range=(1, 2))
X_tfidf_processed = tfidf_vec.fit_transform(df['processed_text'])

# Feature selection using Chi-squared
selector_chi2 = SelectKBest(chi2, k=100)
```

```
X_chi2_processed = selector_chi2.fit_transform(X_tfidf_processed, y_task1

# Get a dense version of our features for models that require it
X_processed_dense = X_tfidf_processed.toarray()

# Initialize k-fold cross validation
k_folds = 10
skf = StratifiedKFold(n_splits=k_folds, shuffle=True, random_state=42)

# Define models with their best parameters based on grid search results
best_models = {
    'Gaussian NB': GaussianNB(var_smoothing=0.0152),
    'Multinomial NB': MultinomialNB(alpha=0.4641, fit_prior=True),
    'Complement NB': ComplementNB(alpha=2.8480, fit_prior=True),
    'Bernoulli NB': BernoulliNB(alpha=0.17475, fit_prior=True),
    'KNN': KNeighborsClassifier(n_neighbors=7, weights='uniform', metric=
    'SVM': SVC(C=1.4174742, kernel='rbf', gamma='scale', probability=True
    'Logistic Regression': LogisticRegression(C=np.float64(4.328761281083
    'Random Forest': RandomForestClassifier(n_estimators=200, max_depth=N
}

# Dictionary to store results
best_cv_results = {}

# Perform k-fold cross validation for each model
for model_name, model in best_models.items():
    print(f"Performing {k_folds}-fold cross-validation for {model_name} w

    # Initialize lists to store performance metrics for each fold
    fold_accuracy = []
    fold_precision = []
    fold_recall = []
    fold_f1 = []
    fold_auc = []

    # For each fold
    for fold, (train_idx, test_idx) in enumerate(skf.split(X_processed_de
        # Split data
        X_train_fold, X_test_fold = X_processed_dense[train_idx], X_proce
        y_train_fold, y_test_fold = y_task1.iloc[train_idx], y_task1.iloc

        # Train model
        model.fit(X_train_fold, y_train_fold)

        # Make predictions
        y_pred_fold = model.predict(X_test_fold)

        # Calculate metrics
        acc = accuracy_score(y_test_fold, y_pred_fold)
        prec = precision_score(y_test_fold, y_pred_fold, zero_division=0)
        rec = recall_score(y_test_fold, y_pred_fold, zero_division=0)
        f1 = f1_score(y_test_fold, y_pred_fold, zero_division=0)

        # For AUC, we need probability estimates
        try:
            if hasattr(model, "predict_proba"):
                y_prob = model.predict_proba(X_test_fold)[:, 1]
                auc_score = roc_auc_score(y_test_fold, y_prob)
                fold_auc.append(auc_score)
        except:
```

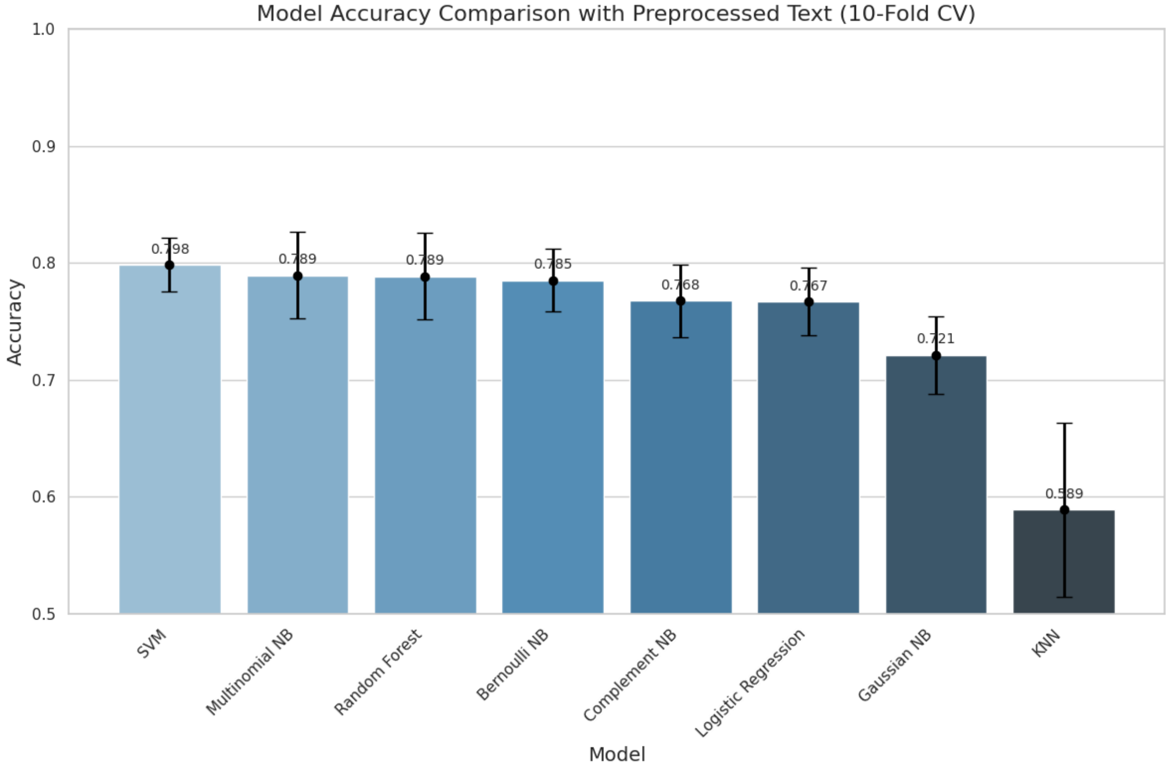
```
                pass

            fold_accuracy.append(acc)
            fold_precision.append(prec)
            fold_recall.append(rec)
            fold_f1.append(f1)

        # Store average metrics and standard deviations
        best_cv_results[model_name] = {
            'accuracy': {
                'mean': np.mean(fold_accuracy),
                'std': np.std(fold_accuracy)
            },
            'precision': {
                'mean': np.mean(fold_precision),
                'std': np.std(fold_precision)
            },
            'recall': {
                'mean': np.mean(fold_recall),
                'std': np.std(fold_recall)
            },
            'f1': {
                'mean': np.mean(fold_f1),
                'std': np.std(fold_f1)
            }
        }

        if fold_auc:
            best_cv_results[model_name]['auc'] = {
                'mean': np.mean(fold_auc),
                'std': np.std(fold_auc)
            }
            print(f"  Average: AUC={best_cv_results[model_name]['auc']['mean'

        print(f"  Average: Accuracy={best_cv_results[model_name]['accuracy'][
        print(f"  Average: F1 Score={best_cv_results[model_name]['f1']['mean'
        print()

# Create DataFrame for visualization
best_results_df = pd.DataFrame({
    'Model': [],
    'Metric': [],
    'Mean': [],
    'Std': []
})

for model_name in best_cv_results:
    for metric in ['accuracy', 'precision', 'recall', 'f1']:
        best_results_df = pd.concat([best_results_df, pd.DataFrame({
            'Model': [model_name],
            'Metric': [metric.capitalize()],
            'Mean': [best_cv_results[model_name][metric]['mean']],
            'Std': [best_cv_results[model_name][metric]['std']]
        })], ignore_index=True)
    if 'auc' in best_cv_results[model_name]:
        best_results_df = pd.concat([best_results_df, pd.DataFrame({
            'Model': [model_name],
            'Metric': ['AUC'],
            'Mean': [best_cv_results[model_name]['auc']['mean']],
            'Std': [best_cv_results[model_name]['auc']['std']]
```

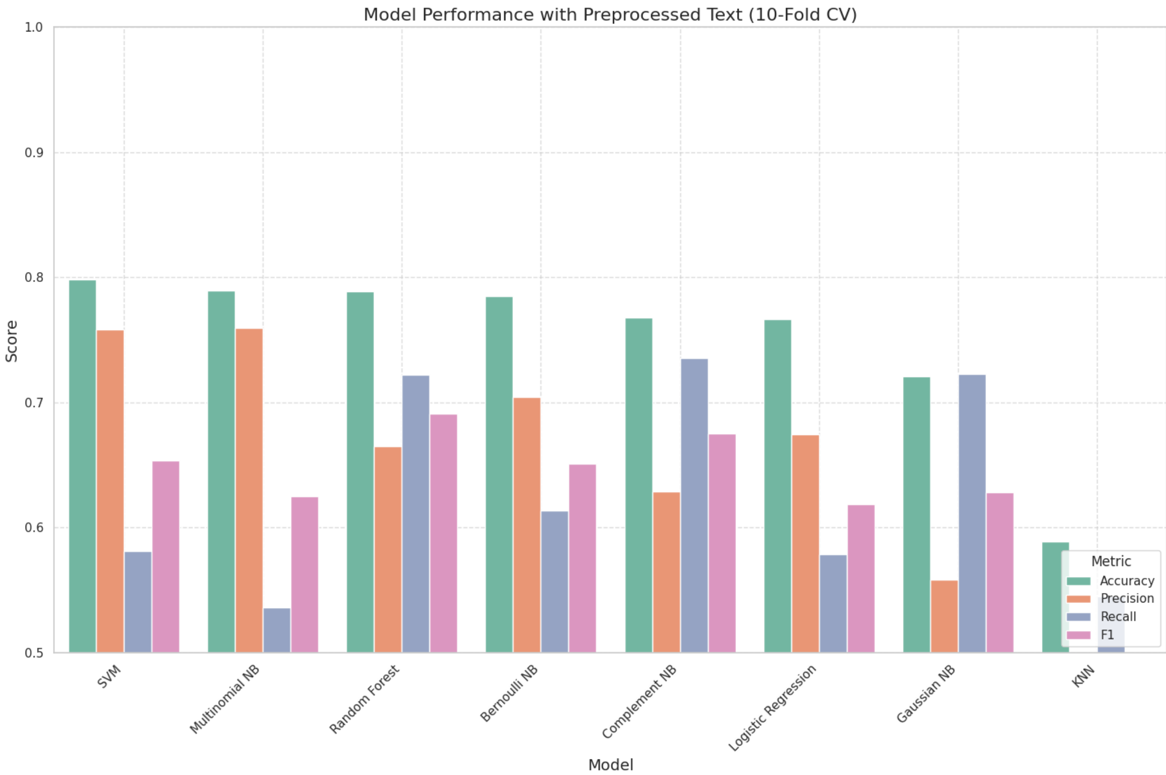
```
        })], ignore_index=True)

# Sort models by accuracy
best_model_order = best_results_df[best_results_df['Metric'] == 'Accuracy

# Create plots
plt.figure(figsize=(12, 8))
sns.set_style("whitegrid")

# Create bar plot for accuracy
ax = sns.barplot(
    data=best_results_df[best_results_df['Metric'] == 'Accuracy'],
    x='Model',
    y='Mean',
    order=best_model_order,
    palette='Blues_d'
)

# Add error bars
for i, model in enumerate(best_model_order):
    row = best_results_df[(best_results_df['Model'] == model) & (best_res
    ax.errorbar(
        i, row['Mean'], yerr=row['Std'],
        fmt='o', color='black', elinewidth=2, capsize=6
    )

# Add value labels on top of bars
for i, bar in enumerate(ax.patches):
    ax.text(
        bar.get_x() + bar.get_width()/2,
        bar.get_height() + 0.01,
        f"{bar.get_height():.3f}",
        ha='center',
        fontsize=10
    )

plt.title(f'Model Accuracy Comparison with Preprocessed Text ({k_folds}-F
plt.xlabel('Model', fontsize=14)
plt.ylabel('Accuracy', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.ylim([0.5, 1.0])
plt.tight_layout()
plt.show()

# Create a grouped bar chart for all metrics
plt.figure(figsize=(15, 10))
sns.set_style("whitegrid")

# Create grouped bar plot
ax = sns.barplot(
    data=best_results_df[best_results_df['Metric'] != 'AUC'],
    x='Model',
    y='Mean',
    hue='Metric',
    order=best_model_order,
    palette='Set2'
)

plt.title(f'Model Performance with Preprocessed Text ({k_folds}-Fold CV)'
plt.xlabel('Model', fontsize=14)
```

```python
plt.ylabel('Score', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.legend(title='Metric', loc='lower right')
plt.ylim([0.5, 1.0])
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
Performing 10-fold cross-validation for Gaussian NB with best parameters o
n preprocessed text...
  Average: AUC=0.7498 ± 0.0386
  Average: Accuracy=0.7211 ± 0.0332
  Average: F1 Score=0.6283 ± 0.0560

Performing 10-fold cross-validation for Multinomial NB with best parameter
s on preprocessed text...
  Average: AUC=0.8442 ± 0.0334
  Average: Accuracy=0.7895 ± 0.0370
  Average: F1 Score=0.6251 ± 0.0685

Performing 10-fold cross-validation for Complement NB with best parameters
on preprocessed text...
  Average: AUC=0.8550 ± 0.0328
  Average: Accuracy=0.7675 ± 0.0310
  Average: F1 Score=0.6750 ± 0.0425

Performing 10-fold cross-validation for Bernoulli NB with best parameters
on preprocessed text...
  Average: AUC=0.8421 ± 0.0246
  Average: Accuracy=0.7851 ± 0.0270
  Average: F1 Score=0.6510 ± 0.0474

Performing 10-fold cross-validation for KNN with best parameters on prepro
cessed text...
  Average: AUC=0.6058 ± 0.0533
  Average: Accuracy=0.5886 ± 0.0745
  Average: F1 Score=0.4461 ± 0.1151

Performing 10-fold cross-validation for SVM with best parameters on prepro
cessed text...
  Average: AUC=0.8463 ± 0.0323
  Average: Accuracy=0.7982 ± 0.0229
  Average: F1 Score=0.6534 ± 0.0402

Performing 10-fold cross-validation for Logistic Regression with best para
meters on preprocessed text...
  Average: AUC=0.8347 ± 0.0318
  Average: Accuracy=0.7667 ± 0.0289
  Average: F1 Score=0.6182 ± 0.0508

Performing 10-fold cross-validation for Random Forest with best parameters
on preprocessed text...
  Average: AUC=0.8573 ± 0.0307
  Average: Accuracy=0.7886 ± 0.0373
  Average: F1 Score=0.6906 ± 0.0609
```

```
/tmp/ipykernel_368067/2418365623.py:144: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.

  ax = sns.barplot(
```

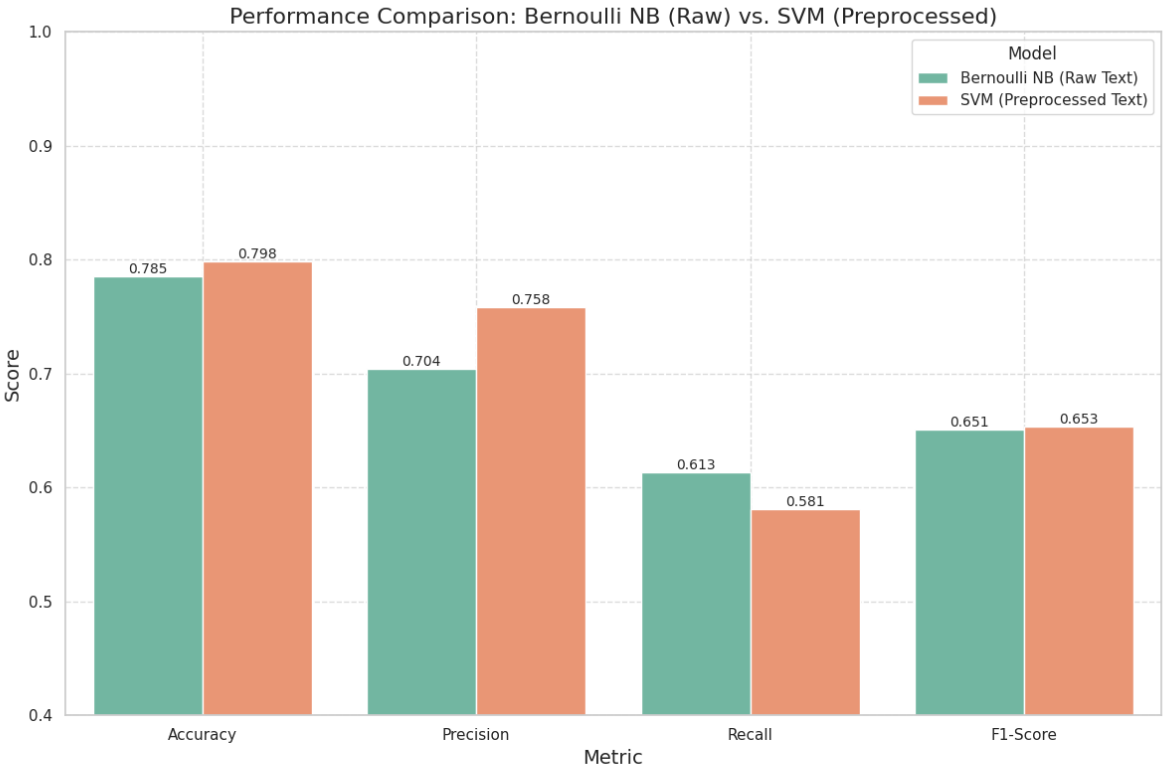




In [99]:
```
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix, roc_curve, auc
```

```python
# Compare performance of SVM with preprocessed data and Bernoulli NB with
import matplotlib.pyplot as plt

# Create a DataFrame for comparison
comparison_data = [
    {
        'Model': 'Bernoulli NB (Raw Text)',
        'Accuracy': best_cv_results['Bernoulli NB']['accuracy']['mean'],
        'Precision': best_cv_results['Bernoulli NB']['precision']['mean']
        'Recall': best_cv_results['Bernoulli NB']['recall']['mean'],
        'F1-Score': best_cv_results['Bernoulli NB']['f1']['mean']
    },
    {
        'Model': 'SVM (Preprocessed Text)',
        'Accuracy': best_cv_results['SVM']['accuracy']['mean'],
        'Precision': best_cv_results['SVM']['precision']['mean'],
        'Recall': best_cv_results['SVM']['recall']['mean'],
        'F1-Score': best_cv_results['SVM']['f1']['mean']
    }
]

# Create DataFrame
compare_df = pd.DataFrame(comparison_data)

# Reshape data for visualization
compare_melted = pd.melt(compare_df, id_vars='Model',
                         value_vars=['Accuracy', 'Precision', 'Recall', '
                         var_name='Metric', value_name='Score')

# Plot comparison
plt.figure(figsize=(12, 8))
sns.set_style("whitegrid")

# Create grouped bar chart
ax = sns.barplot(data=compare_melted, x='Metric', y='Score', hue='Model',

plt.title('Performance Comparison: Bernoulli NB (Raw) vs. SVM (Preprocess
plt.xlabel('Metric', fontsize=14)
plt.ylabel('Score', fontsize=14)
plt.ylim([0.4, 1.0])
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(title='Model')

# Add value labels on bars
for p in ax.patches:
    ax.annotate(f'{p.get_height():.3f}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'bottom',
                fontsize=10)

plt.tight_layout()
plt.show()

# Statistical significance test (if applicable)
print("\nPerformance Summary:")
print(compare_df.set_index('Model'))

# Analyze differences
diff_accuracy = abs(comparison_data[0]['Accuracy'] - comparison_data[1]['
diff_f1 = abs(comparison_data[0]['F1-Score'] - comparison_data[1]['F1-Sco
```

```python
print(f"\nDifference in Accuracy: {diff_accuracy:.3f}")
print(f"Difference in F1-Score: {diff_f1:.3f}")

# Print conclusions
if comparison_data[0]['Accuracy'] > comparison_data[1]['Accuracy']:
    winner = "Bernoulli NB with raw text"
else:
    winner = "SVM with preprocessed text"

print(f"\nConclusion: {winner} performs better in terms of accuracy.")
print("This suggests that " +
      ("text preprocessing is beneficial for this classification task."
       if winner == "SVM with preprocessed text"
       else "raw text features might contain important signals that are l
```




```
Performance Summary:
                         Accuracy  Precision    Recall  F1-Score
Model
Bernoulli NB (Raw Text)  0.785088   0.704412  0.613158  0.651009
SVM (Preprocessed Text)  0.798246   0.758331  0.581081  0.653448

Difference in Accuracy: 0.013
Difference in F1-Score: 0.002

Conclusion: SVM with preprocessed text performs better in terms of accurac
y.
This suggests that text preprocessing is beneficial for this classificatio
n task.
```

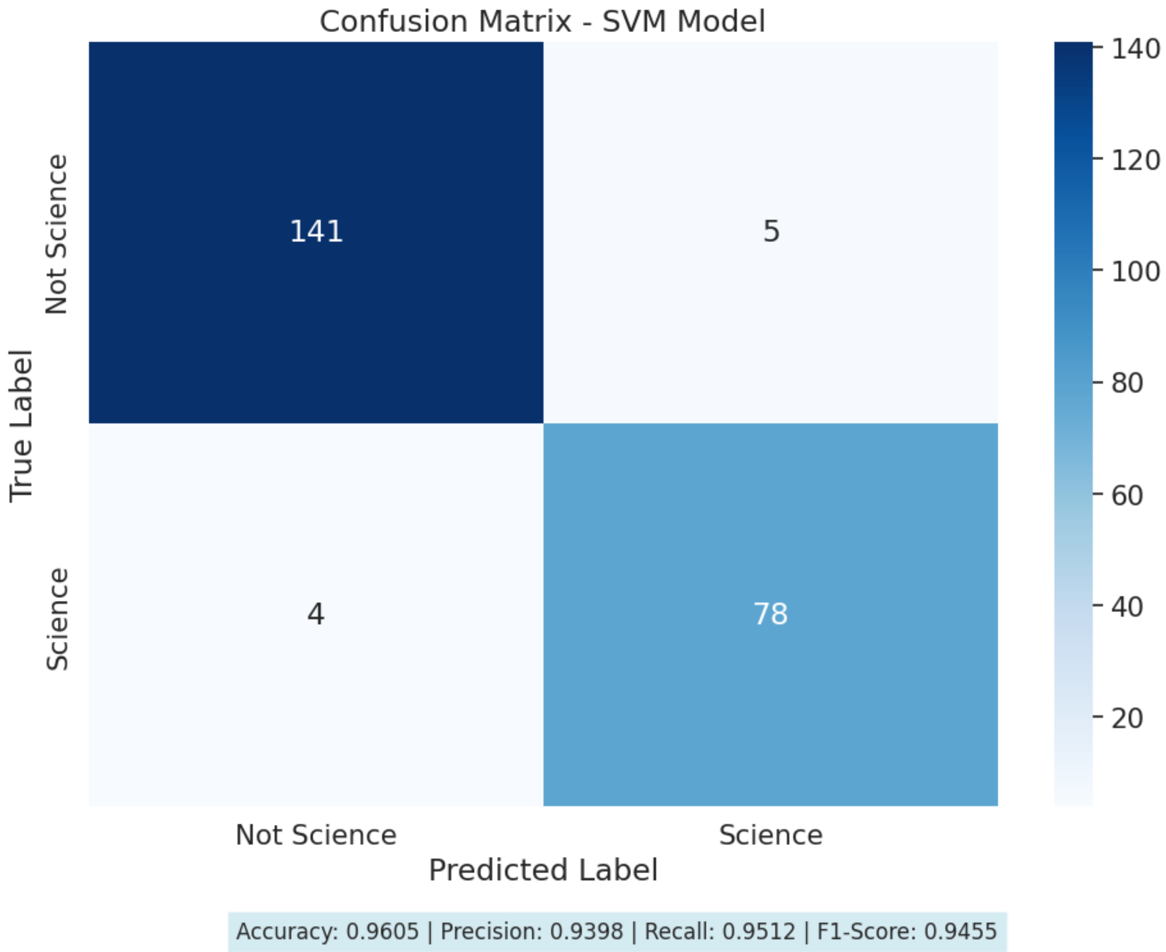
In [100…

```python
best_model = best_models['SVM']
```

In [101…

```python
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import numpy as np

import matplotlib.pyplot as plt
```

```python
# Get predictions from the best model on the test set
y_pred = best_model.predict(X_test_task1)

# Calculate the confusion matrix
cm = confusion_matrix(y_test_task1, y_pred)

# Create a prettier visualization of the confusion matrix
plt.figure(figsize=(10, 8))
sns.set(font_scale=1.4)
ax = sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                 xticklabels=['Not Science', 'Science'],
                 yticklabels=['Not Science', 'Science'])

# Add labels, title and ticks
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix - SVM Model')

# Calculate metrics
report = classification_report(y_test_task1, y_pred, output_dict=True)
accuracy = (cm[0, 0] + cm[1, 1]) / np.sum(cm)
precision = report['1']['precision']
recall = report['1']['recall']
f1 = report['1']['f1-score']

# Display metrics on the plot
plt.figtext(0.5, 0.01, f'Accuracy: {accuracy:.4f} | Precision: {precision
            ha='center', fontsize=12, bbox={'facecolor': 'lightblue', 'al

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



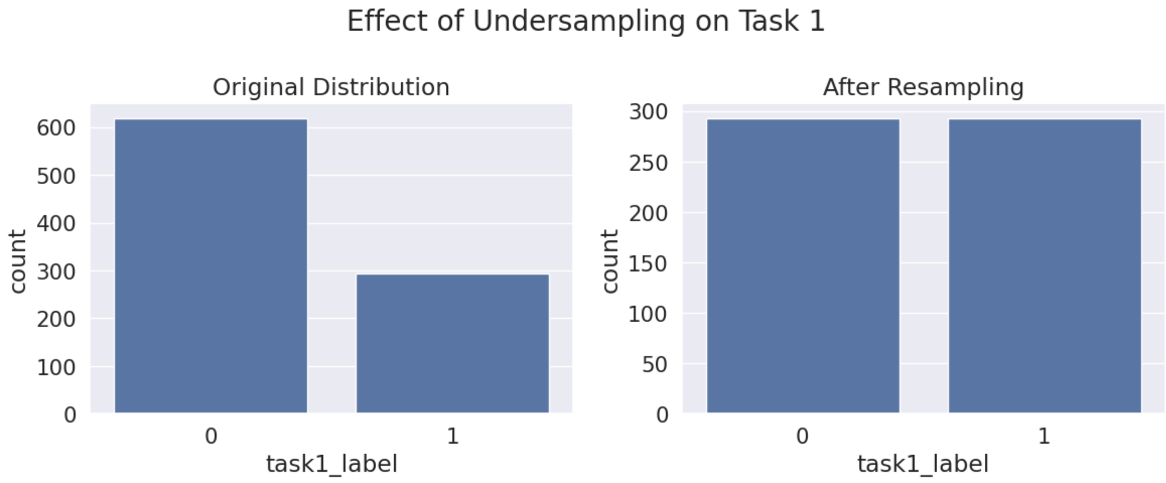
In [102…
```
%pip install imbalanced-learn

# Import required modules
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler, SMOTE
from collections import Counter
```

```
Requirement already satisfied: imbalanced-learn in ./.venv/lib/python3.10/
site-packages (0.13.0)
Requirement already satisfied: numpy<3,>=1.24.3 in ./.venv/lib/python3.10/
site-packages (from imbalanced-learn) (2.2.5)
Requirement already satisfied: scipy<2,>=1.10.1 in ./.venv/lib/python3.10/
site-packages (from imbalanced-learn) (1.15.2)
Requirement already satisfied: scikit-learn<2,>=1.3.2 in ./.venv/lib/pytho
n3.10/site-packages (from imbalanced-learn) (1.6.1)
Requirement already satisfied: sklearn-compat<1,>=0.1 in ./.venv/lib/pytho
n3.10/site-packages (from imbalanced-learn) (0.1.3)
Requirement already satisfied: joblib<2,>=1.1.1 in ./.venv/lib/python3.10/
site-packages (from imbalanced-learn) (1.4.2)
Requirement already satisfied: threadpoolctl<4,>=2.0.0 in ./.venv/lib/pyth
on3.10/site-packages (from imbalanced-learn) (3.6.0)
Note: you may need to restart the kernel to use updated packages.
```

In [103…
```
# For Task 1 (science_related classification)
def apply_undersampling(X, y):
    undersampler = RandomUnderSampler(random_state=42)
    X_resampled, y_resampled = undersampler.fit_resample(X, y)
    print(f"Original distribution: {Counter(y)}")
    print(f"Distribution after undersampling: {Counter(y_resampled)}")
    return X_resampled, y_resampled

# Apply to Task 1
X_train_task1_under, y_train_task1_under = apply_undersampling(X_train_ta
```

```
Original distribution: Counter({0: 619, 1: 293})
Distribution after undersampling: Counter({0: 293, 1: 293})
```

In [104…
```
# Visualize effect of sampling
def plot_sampling_comparison(original_y, resampled_y, title):
    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    sns.countplot(x=original_y)
    plt.title('Original Distribution')

    plt.subplot(1, 2, 2)
    sns.countplot(x=resampled_y)
    plt.title('After Resampling')

    plt.suptitle(title)
    plt.tight_layout()
    plt.show()

# Example for Task 1
plot_sampling_comparison(y_train_task1, y_train_task1_under, 'Effect of U
```

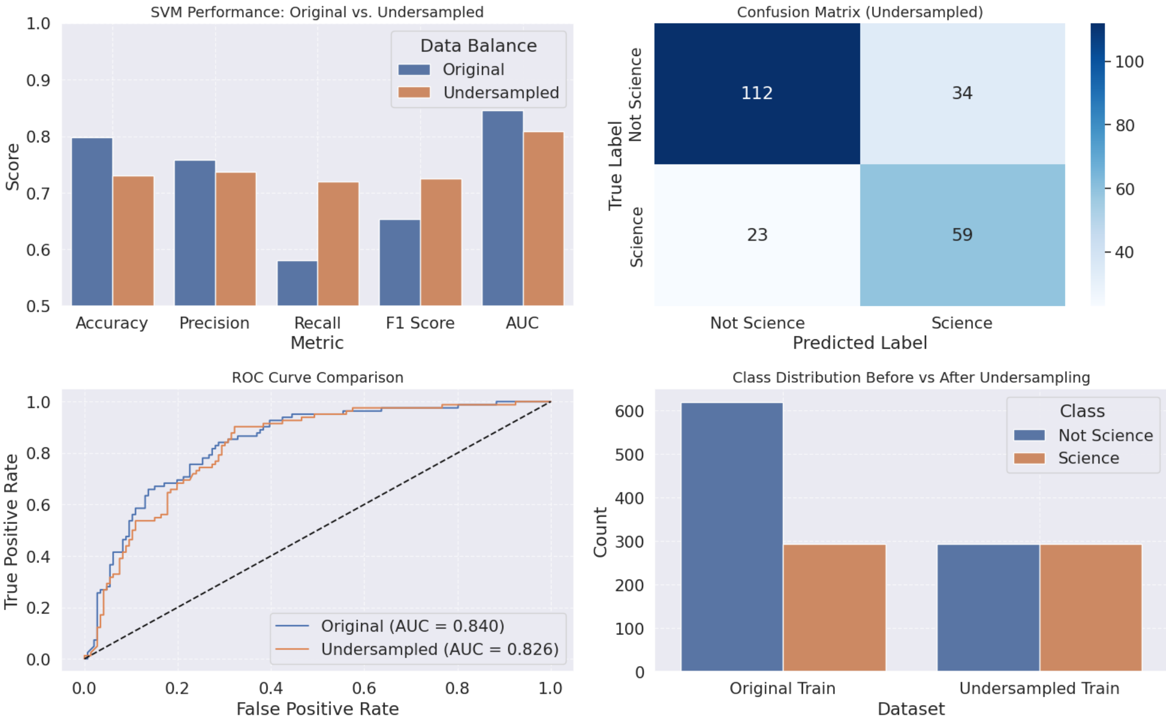


In [105…

```
# Retrain the best model (SVM) with undersampled data
best_model = best_models['SVM']

# Evaluate on the original test data for fair comparison
best_model.fit(X_train_task1_under, y_train_task1_under)
y_pred_under = best_model.predict(X_test_task1)
accuracy_under = accuracy_score(y_test_task1, y_pred_under)
report_under = classification_report(y_test_task1, y_pred_under, output_d
cm_under = confusion_matrix(y_test_task1, y_pred_under)

# Perform k-fold cross validation with undersampled data
# Initialize StratifiedKFold
skf = StratifiedKFold(n_splits=k_folds, shuffle=True, random_state=42)

# Store evaluation metrics
fold_accuracy_under = []
fold_precision_under = []
fold_recall_under = []
fold_f1_under = []
fold_auc_under = []

# Get a balanced dataset
X_balanced = X_train_task1_under
y_balanced = y_train_task1_under

# Perform cross-validation
for fold, (train_idx, test_idx) in enumerate(skf.split(X_balanced, y_bala
    X_train_fold, X_test_fold = X_balanced[train_idx], X_balanced[test_id
    y_train_fold, y_test_fold = y_balanced.iloc[train_idx], y_balanced.il

    best_model.fit(X_train_fold, y_train_fold)
    y_pred_fold = best_model.predict(X_test_fold)

    # Calculate metrics
    acc = accuracy_score(y_test_fold, y_pred_fold)
    prec = precision_score(y_test_fold, y_pred_fold, zero_division=0)
    rec = recall_score(y_test_fold, y_pred_fold, zero_division=0)
    f1 = f1_score(y_test_fold, y_pred_fold, zero_division=0)

    # For AUC
    if hasattr(best_model, "predict_proba"):
        y_prob = best_model.predict_proba(X_test_fold)[:, 1]
        auc_score = roc_auc_score(y_test_fold, y_prob)
        fold_auc_under.append(auc_score)
```

```
    fold_accuracy_under.append(acc)
    fold_precision_under.append(prec)
    fold_recall_under.append(rec)
    fold_f1_under.append(f1)

# Compare performance metrics before and after undersampling
comparison_data = {
    'Metric': ['Accuracy', 'Precision', 'Recall', 'F1 Score', 'AUC'],
    'Original': [
        best_cv_results['SVM']['accuracy']['mean'],
        best_cv_results['SVM']['precision']['mean'],
        best_cv_results['SVM']['recall']['mean'],
        best_cv_results['SVM']['f1']['mean'],
        best_cv_results['SVM']['auc']['mean'] if 'auc' in best_cv_results
    ],
    'Undersampled': [
        np.mean(fold_accuracy_under),
        np.mean(fold_precision_under),
        np.mean(fold_recall_under),
        np.mean(fold_f1_under),
        np.mean(fold_auc_under) if fold_auc_under else None
    ]
}

comparison_df = pd.DataFrame(comparison_data)

# Create visualizations
plt.figure(figsize=(16, 10))

# Bar chart comparison
plt.subplot(2, 2, 1)
comparison_melted = pd.melt(comparison_df, id_vars=['Metric'],
                           value_vars=['Original', 'Undersampled'],
                           var_name='Data Balance', value_name='Score')

# Filter out any None values
comparison_melted = comparison_melted.dropna()

sns.barplot(x='Metric', y='Score', hue='Data Balance', data=comparison_me
plt.title('SVM Performance: Original vs. Undersampled', fontsize=14)
plt.ylim(0.5, 1.0)
plt.grid(True, linestyle='--', alpha=0.7)

# Confusion matrix for undersampled model
plt.subplot(2, 2, 2)
sns.heatmap(cm_under, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Not Science', 'Science'],
            yticklabels=['Not Science', 'Science'])
plt.title('Confusion Matrix (Undersampled)', fontsize=14)
plt.ylabel('True Label')
plt.xlabel('Predicted Label')

# ROC Curve comparison if available
if hasattr(best_model, "predict_proba"):
    plt.subplot(2, 2, 3)
    # For original model
    y_prob_orig = best_model.fit(X_train_task1, y_train_task1).predict_pr
    fpr_orig, tpr_orig, _ = roc_curve(y_test_task1, y_prob_orig)
    auc_orig = auc(fpr_orig, tpr_orig)
```

```
    # For undersampled model
    best_model.fit(X_train_task1_under, y_train_task1_under)
    y_prob_under = best_model.predict_proba(X_test_task1)[:, 1]
    fpr_under, tpr_under, _ = roc_curve(y_test_task1, y_prob_under)
    auc_under = auc(fpr_under, tpr_under)

    # Plot ROC curves
    plt.plot(fpr_orig, tpr_orig, label=f'Original (AUC = {auc_orig:.3f})'
    plt.plot(fpr_under, tpr_under, label=f'Undersampled (AUC = {auc_under
    plt.plot([0, 1], [0, 1], 'k--')
    plt.title('ROC Curve Comparison', fontsize=14)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.legend()
    plt.grid(True, linestyle='--', alpha=0.7)

# Class distribution before and after undersampling
plt.subplot(2, 2, 4)
class_dist = pd.DataFrame({
    'Dataset': ['Original Train'] * 2 + ['Undersampled Train'] * 2,
    'Class': ['Not Science', 'Science'] * 2,
    'Count': [
        sum(y_train_task1 == 0),
        sum(y_train_task1 == 1),
        sum(y_train_task1_under == 0),
        sum(y_train_task1_under == 1)
    ]
})
sns.barplot(x='Dataset', y='Count', hue='Class', data=class_dist)
plt.title('Class Distribution Before vs After Undersampling', fontsize=14
plt.grid(True, linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()

# Print summary
print("\nSVM Performance Comparison Summary:")
print(comparison_df.set_index('Metric'))
print("\nOriginal Class Distribution:", Counter(y_train_task1))
print("Undersampled Class Distribution:", Counter(y_train_task1_under))

# Calculate improvement percentages
improvement_df = pd.DataFrame({
    'Metric': comparison_df['Metric'],
    'Improvement (%)': [
        ((comparison_df['Undersampled'][i] - comparison_df['Original'][i]
        if comparison_df['Original'][i] is not None and comparison_df['Un
        else None
        for i in range(len(comparison_df))
    ]
})
print("\nRelative Improvement After Undersampling:")
print(improvement_df.set_index('Metric'))
```




```
SVM Performance Comparison Summary:
           Original  Undersampled
Metric
Accuracy   0.798246      0.730479
Precision  0.758331      0.737308
Recall     0.581081      0.719885
F1 Score   0.653448      0.725460
AUC        0.846289      0.809384

Original Class Distribution: Counter({0: 619, 1: 293})
Undersampled Class Distribution: Counter({0: 293, 1: 293})

Relative Improvement After Undersampling:
           Improvement (%)
Metric
Accuracy         -8.489412
Precision        -2.772259
Recall           23.887196
F1 Score         11.020415
AUC              -4.360885
```

In [123…

```
# TASK 2: Scientific Claim/Reference Classification

# Since Task 2 is only applicable to science-related tweets, we need to u
print("Task 2: Scientific Claim/Reference Classification\n" + "="*50)

# Get a dense version of features for science-related tweets
# Convert the pandas Series to a numpy array with .values
X_dense_sci = X_selected[(df['science_related'] == 1).values].toarray()
y_task2 = df_sci['task2_label']

# Split data for Task 2
X_train_task2, X_test_task2, y_train_task2, y_test_task2 = train_test_spl
    X_dense_sci, y_task2, test_size=0.2, random_state=42
)

# Define the best models from Task 1 (you can modify these based on Task
task2_models = {
```

```
    'Gaussian NB': GaussianNB(var_smoothing=0.0152),
    'Multinomial NB': MultinomialNB(alpha=0.4641, fit_prior=True),
    'Complement NB': ComplementNB(alpha=2.8480, fit_prior=True),
    'Bernoulli NB': BernoulliNB(alpha=0.17475, fit_prior=True),
    'KNN': KNeighborsClassifier(n_neighbors=7, weights='uniform', metric=
    'SVM': SVC(C=1.4174742, kernel='rbf', gamma='scale', probability=True
    'Logistic Regression': LogisticRegression(C=np.float64(4.328761281083
    'Random Forest': RandomForestClassifier(n_estimators=200, max_depth=N
}

# Train and evaluate models for Task 2
task2_results = {}
for name, model in task2_models.items():
    task2_results[name] = evaluate_model(
        model, X_train_task2, X_test_task2, y_train_task2, y_test_task2,
    )
    print("\n")
```

```
Task 2: Scientific Claim/Reference Classification
==================================================
Model: Gaussian NB
Accuracy: 0.8533
              precision    recall  f1-score   support

           0       0.30      0.43      0.35         7
           1       0.94      0.90      0.92        68

    accuracy                           0.85        75
   macro avg       0.62      0.66      0.64        75
weighted avg       0.88      0.85      0.86        75



Model: Multinomial NB
Accuracy: 0.9067
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         7
           1       0.91      1.00      0.95        68

    accuracy                           0.91        75
   macro avg       0.45      0.50      0.48        75
weighted avg       0.82      0.91      0.86        75



Model: Complement NB
Accuracy: 0.7200
              precision    recall  f1-score   support

           0       0.18      0.57      0.28         7
           1       0.94      0.74      0.83        68

    accuracy                           0.72        75
   macro avg       0.56      0.65      0.55        75
weighted avg       0.87      0.72      0.78        75



Model: Bernoulli NB
Accuracy: 0.8533
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         7
           1       0.90      0.94      0.92        68

    accuracy                           0.85        75
   macro avg       0.45      0.47      0.46        75
weighted avg       0.82      0.85      0.83        75



Model: KNN
Accuracy: 0.9067
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         7
           1       0.91      1.00      0.95        68
```

```
    accuracy                           0.91        75
   macro avg       0.45      0.50      0.48        75
weighted avg       0.82      0.91      0.86        75



Model: SVM
Accuracy: 0.9200
              precision    recall  f1-score   support

           0       1.00      0.14      0.25         7
           1       0.92      1.00      0.96        68

    accuracy                           0.92        75
   macro avg       0.96      0.57      0.60        75
weighted avg       0.93      0.92      0.89        75



Model: Logistic Regression
Accuracy: 0.9200
              precision    recall  f1-score   support

           0       1.00      0.14      0.25         7
           1       0.92      1.00      0.96        68

    accuracy                           0.92        75
   macro avg       0.96      0.57      0.60        75
weighted avg       0.93      0.92      0.89        75



Model: Random Forest
Accuracy: 0.9200
              precision    recall  f1-score   support

           0       1.00      0.14      0.25         7
           1       0.92      1.00      0.96        68

    accuracy                           0.92        75
   macro avg       0.96      0.57      0.60        75
weighted avg       0.93      0.92      0.89        75
```

In [107…

```python
# Cross-validation for Task 2
print("Task 2: Cross-Validation\n" + "="*40)

# Initialize k-fold cross validation
k_folds = 10
skf = StratifiedKFold(n_splits=k_folds, shuffle=True, random_state=42)

# Dictionary to store CV results
task2_cv_results = {}

# Perform k-fold cross validation for each model
for model_name, model in task2_models.items():
    print(f"Performing {k_folds}-fold cross-validation for {model_name}..
```

```
# Initialize lists to store performance metrics for each fold
fold_accuracy = []
fold_precision = []
fold_recall = []
fold_f1 = []
fold_auc = []

# For each fold
for fold, (train_idx, test_idx) in enumerate(skf.split(X_dense_sci, y
    # Split data
    X_train_fold, X_test_fold = X_dense_sci[train_idx], X_dense_sci[t
    y_train_fold, y_test_fold = y_task2.iloc[train_idx], y_task2.iloc

    # Train model
    model.fit(X_train_fold, y_train_fold)

    # Make predictions
    y_pred_fold = model.predict(X_test_fold)

    # Calculate metrics
    acc = accuracy_score(y_test_fold, y_pred_fold)
    prec = precision_score(y_test_fold, y_pred_fold, zero_division=0)
    rec = recall_score(y_test_fold, y_pred_fold, zero_division=0)
    f1 = f1_score(y_test_fold, y_pred_fold, zero_division=0)

    # For AUC, we need probability estimates
    try:
        if hasattr(model, "predict_proba"):
            y_prob = model.predict_proba(X_test_fold)[:, 1]
            auc_score = roc_auc_score(y_test_fold, y_prob)
            fold_auc.append(auc_score)
    except:
        pass

    fold_accuracy.append(acc)
    fold_precision.append(prec)
    fold_recall.append(rec)
    fold_f1.append(f1)

# Store average metrics and standard deviations
task2_cv_results[model_name] = {
    'accuracy': {
        'mean': np.mean(fold_accuracy),
        'std': np.std(fold_accuracy)
    },
    'precision': {
        'mean': np.mean(fold_precision),
        'std': np.std(fold_precision)
    },
    'recall': {
        'mean': np.mean(fold_recall),
        'std': np.std(fold_recall)
    },
    'f1': {
        'mean': np.mean(fold_f1),
        'std': np.std(fold_f1)
    }
}
```

```
    if fold_auc:
        task2_cv_results[model_name]['auc'] = {
            'mean': np.mean(fold_auc),
            'std': np.std(fold_auc)
        }
        print(f"  Average: AUC={task2_cv_results[model_name]['auc']['mean

    print(f"  Average: Accuracy={task2_cv_results[model_name]['accuracy']
    print(f"  Average: F1 Score={task2_cv_results[model_name]['f1']['mean
    print()
```

```
Task 2: Cross-Validation
========================================
Performing 10-fold cross-validation for Gaussian NB...
  Average: AUC=0.5660 ± 0.1187
  Average: Accuracy=0.8217 ± 0.0635
  Average: F1 Score=0.8988 ± 0.0375

Performing 10-fold cross-validation for Multinomial NB...
  Average: AUC=0.7806 ± 0.0932
  Average: Accuracy=0.9148 ± 0.0152
  Average: F1 Score=0.9554 ± 0.0081

Performing 10-fold cross-validation for Complement NB...
  Average: AUC=0.7468 ± 0.0979
  Average: Accuracy=0.7151 ± 0.0768
  Average: F1 Score=0.8205 ± 0.0532

Performing 10-fold cross-validation for Bernoulli NB...
  Average: AUC=0.7860 ± 0.0991
  Average: Accuracy=0.8935 ± 0.0332
  Average: F1 Score=0.9423 ± 0.0184

Performing 10-fold cross-validation for KNN...
  Average: AUC=0.5324 ± 0.0660
  Average: Accuracy=0.9094 ± 0.0126
  Average: F1 Score=0.9525 ± 0.0069

Performing 10-fold cross-validation for SVM...
  Average: AUC=0.7151 ± 0.1219
  Average: Accuracy=0.9175 ± 0.0179
  Average: F1 Score=0.9567 ± 0.0095

Performing 10-fold cross-validation for Logistic Regression...
  Average: AUC=0.7019 ± 0.1686
  Average: Accuracy=0.9068 ± 0.0242
  Average: F1 Score=0.9503 ± 0.0131

Performing 10-fold cross-validation for Random Forest...
  Average: AUC=0.7275 ± 0.1402
  Average: Accuracy=0.9148 ± 0.0152
  Average: F1 Score=0.9554 ± 0.0081
```

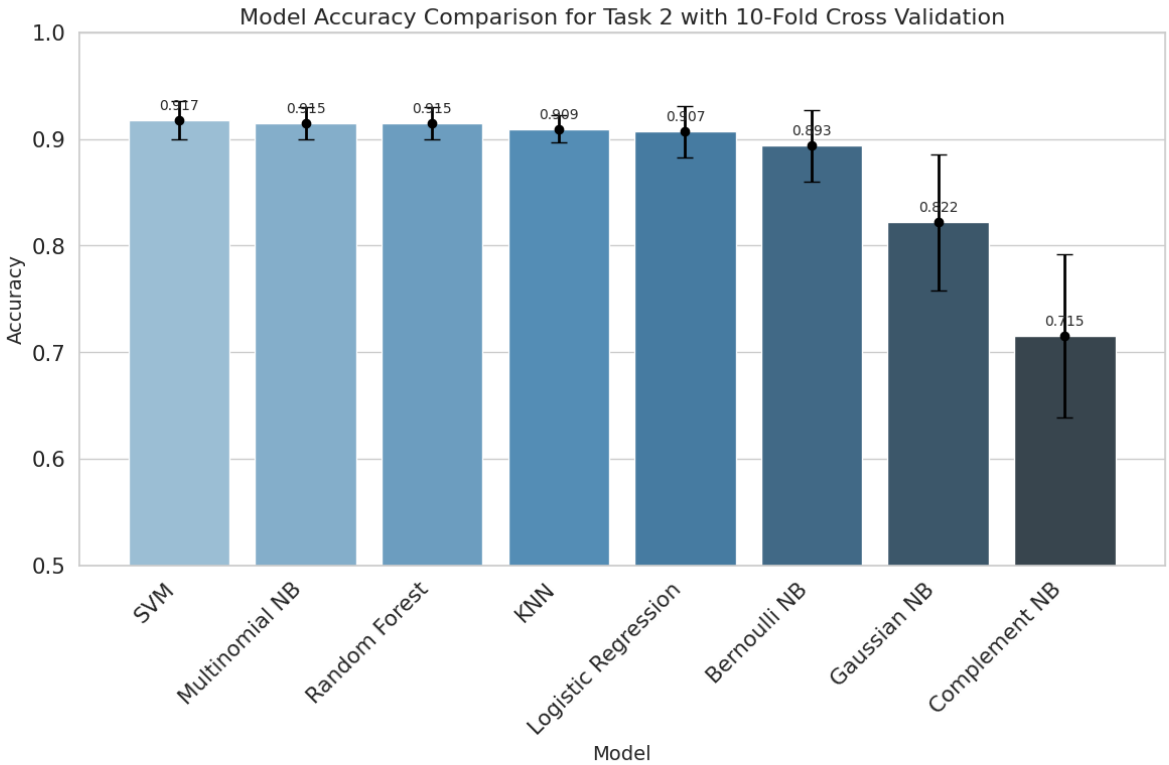
In [108…

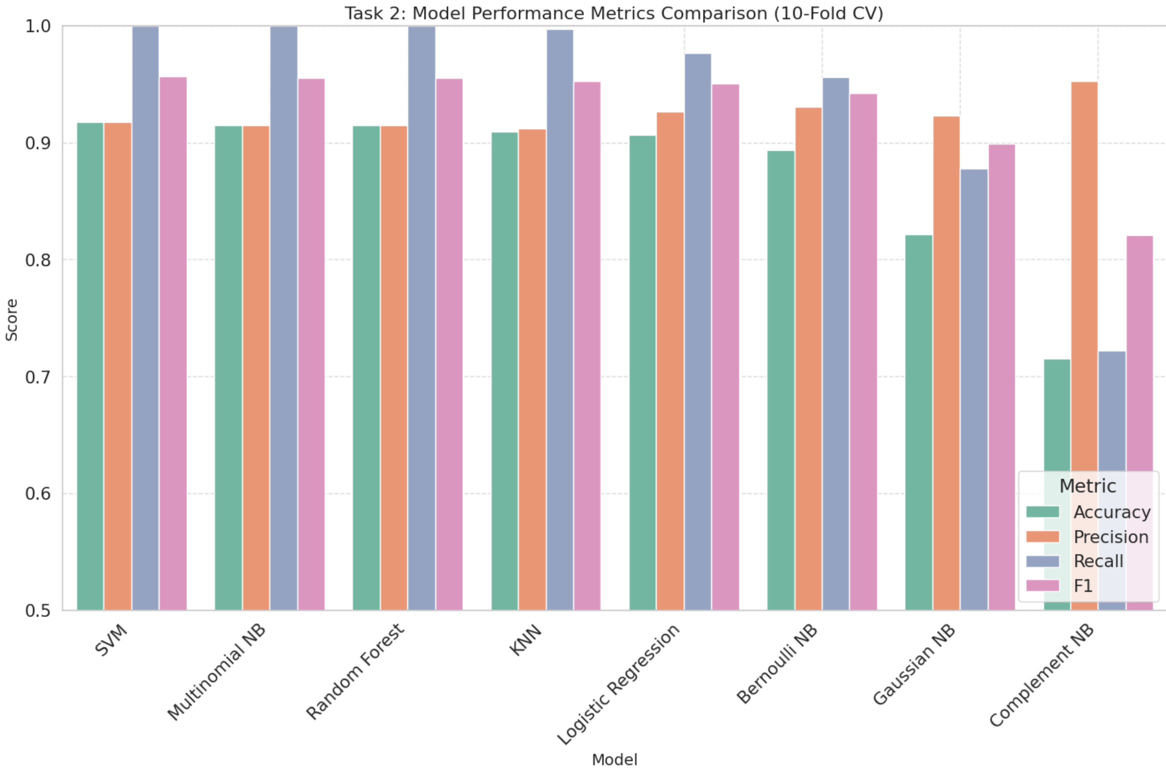
```
# Visualize Task 2 cross-validation results
# Create DataFrame for visualization
task2_results_df = pd.DataFrame({
    'Model': [],
    'Metric': [],
    'Mean': [],
```

```
    'Std': []
})

for model_name in task2_cv_results:
    for metric in ['accuracy', 'precision', 'recall', 'f1']:
        task2_results_df = pd.concat([task2_results_df, pd.DataFrame({
            'Model': [model_name],
            'Metric': [metric.capitalize()],
            'Mean': [task2_cv_results[model_name][metric]['mean']],
            'Std': [task2_cv_results[model_name][metric]['std']]
        })], ignore_index=True)
    if 'auc' in task2_cv_results[model_name]:
        task2_results_df = pd.concat([task2_results_df, pd.DataFrame({
            'Model': [model_name],
            'Metric': ['AUC'],
            'Mean': [task2_cv_results[model_name]['auc']['mean']],
            'Std': [task2_cv_results[model_name]['auc']['std']]
        })], ignore_index=True)

# Sort models by accuracy
task2_model_order = task2_results_df[task2_results_df['Metric'] == 'Accur

# Create accuracy bar plot
plt.figure(figsize=(12, 8))
sns.set_style("whitegrid")

# Create bar plot for accuracy
ax = sns.barplot(
    data=task2_results_df[task2_results_df['Metric'] == 'Accuracy'],
    x='Model',
    y='Mean',
    order=task2_model_order,
    palette='Blues_d'
)

# Add error bars
for i, model in enumerate(task2_model_order):
    row = task2_results_df[(task2_results_df['Model'] == model) & (task2_
    ax.errorbar(
        i, row['Mean'], yerr=row['Std'],
        fmt='o', color='black', elinewidth=2, capsize=6
    )

# Add value labels on top of bars
for i, bar in enumerate(ax.patches):
    ax.text(
        bar.get_x() + bar.get_width()/2,
        bar.get_height() + 0.01,
        f"{bar.get_height():.3f}",
        ha='center',
        fontsize=10
    )

plt.title(f'Model Accuracy Comparison for Task 2 with {k_folds}-Fold Cros
plt.xlabel('Model', fontsize=14)
plt.ylabel('Accuracy', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.ylim([0.5, 1.0])
plt.tight_layout()
plt.show()
```

```
# Create a grouped bar chart for all metrics
plt.figure(figsize=(15, 10))
sns.set_style("whitegrid")

# Create grouped bar plot
ax = sns.barplot(
    data=task2_results_df[task2_results_df['Metric'] != 'AUC'],
    x='Model',
    y='Mean',
    hue='Metric',
    order=task2_model_order,
    palette='Set2'
)

plt.title(f'Task 2: Model Performance Metrics Comparison ({k_folds}-Fold
plt.xlabel('Model', fontsize=14)
plt.ylabel('Score', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.legend(title='Metric', loc='lower right')
plt.ylim([0.5, 1.0])
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
/tmp/ipykernel_368067/1744879752.py:34: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.

  ax = sns.barplot(
```



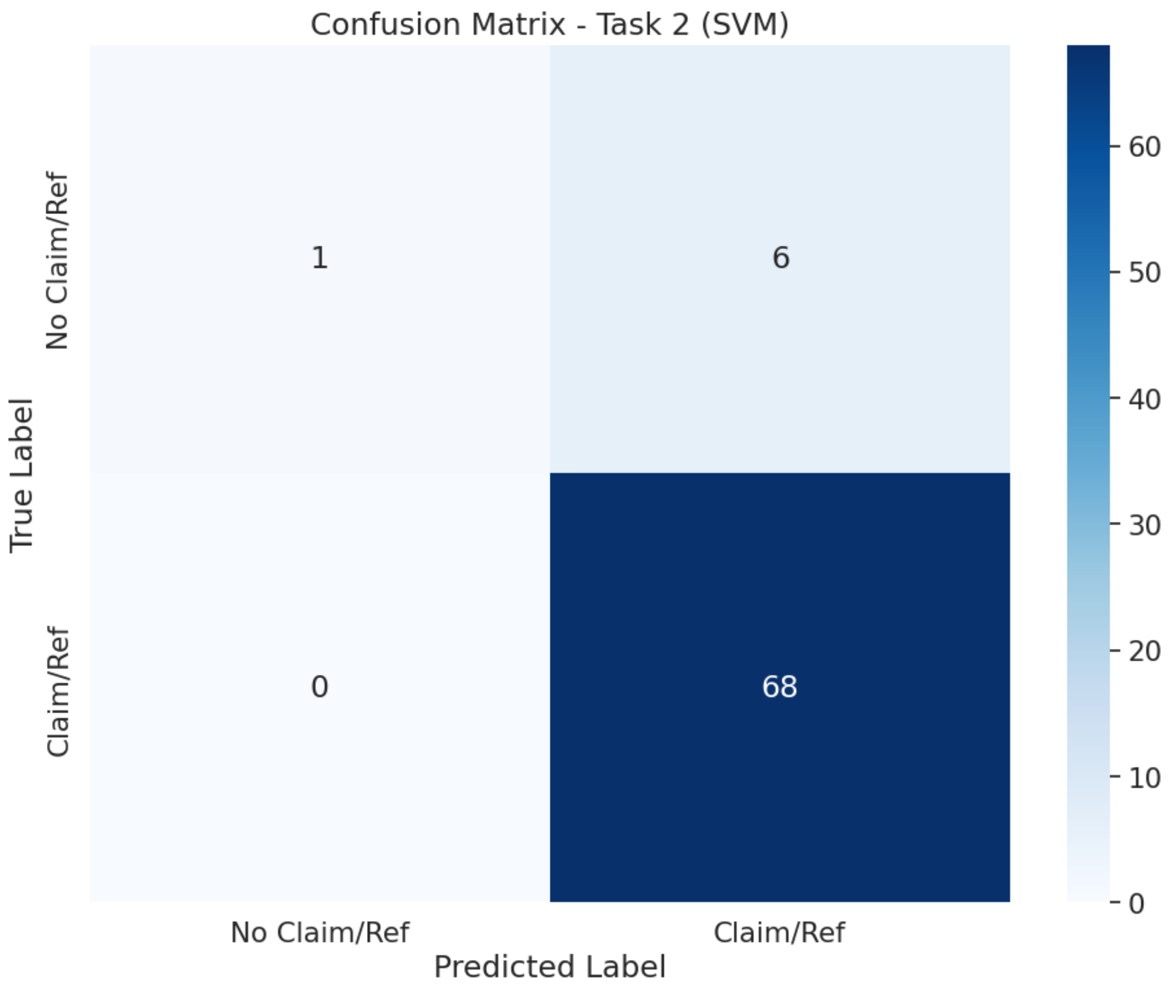


In [109…

```
# Hyperparameter optimization for the best model on Task 2
# Let's assume SVM was the best model (update based on your results)
best_task2_model_name = 'SVM'  # Replace with the actual best model from
best_task2_model = task2_models[best_task2_model_name]

print(f"Performing Grid Search for {best_task2_model_name} on Task 2...")

# Define the parameter grid for the best model
if best_task2_model_name == 'SVM':
    param_grid = {
        'C': np.logspace(-3, 3, 10),
        'gamma': ['scale', 'auto'] + list(np.logspace(-3, 3, 5)),
        'kernel': ['rbf', 'linear']
    }
elif best_task2_model_name == 'Random Forest':
    param_grid = {
        'n_estimators': [50, 100, 200],
        'max_depth': [None, 10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }
elif best_task2_model_name.endswith('NB'):
    if best_task2_model_name == 'Gaussian NB':
        param_grid = {
            'var_smoothing': np.logspace(-10, 0, 11)
        }
    else:
        param_grid = {
            'alpha': np.logspace(-3, 3, 10),
            'fit_prior': [True, False]
        }
else:
    # Default grid for other models
    param_grid = {
        'C': np.logspace(-3, 3, 10)
    }
```

```python
# Create and fit GridSearchCV
grid_search_task2 = GridSearchCV(
    estimator=best_task2_model,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=1
)

grid_search_task2.fit(X_train_task2, y_train_task2)

# Display the best parameters and score
print(f"Best Parameters: {grid_search_task2.best_params_}")
print(f"Best Cross-Validation Score: {grid_search_task2.best_score_:.4f}"

# Evaluate on test set
best_model_task2 = grid_search_task2.best_estimator_
y_pred_task2 = best_model_task2.predict(X_test_task2)
accuracy_task2 = accuracy_score(y_test_task2, y_pred_task2)
print(f"Test Accuracy with Best Parameters: {accuracy_task2:.4f}")
print("\nClassification Report:")
print(classification_report(y_test_task2, y_pred_task2))

# Plot confusion matrix
plt.figure(figsize=(10, 8))
cm = confusion_matrix(y_test_task2, y_pred_task2)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['No Claim/Ref', 'Claim/Ref'],
            yticklabels=['No Claim/Ref', 'Claim/Ref'])
plt.title(f'Confusion Matrix - Task 2 ({best_task2_model_name})')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.tight_layout()
plt.show()
```

```
Performing Grid Search for SVM on Task 2...
Fitting 5 folds for each of 140 candidates, totalling 700 fits
Best Parameters: {'C': np.float64(215.44346900318823), 'gamma': 'auto', 'k
ernel': 'rbf'}
Best Cross-Validation Score: 0.9167
Test Accuracy with Best Parameters: 0.9200

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.14      0.25         7
           1       0.92      1.00      0.96        68

    accuracy                           0.92        75
   macro avg       0.96      0.57      0.60        75
weighted avg       0.93      0.92      0.89        75
```

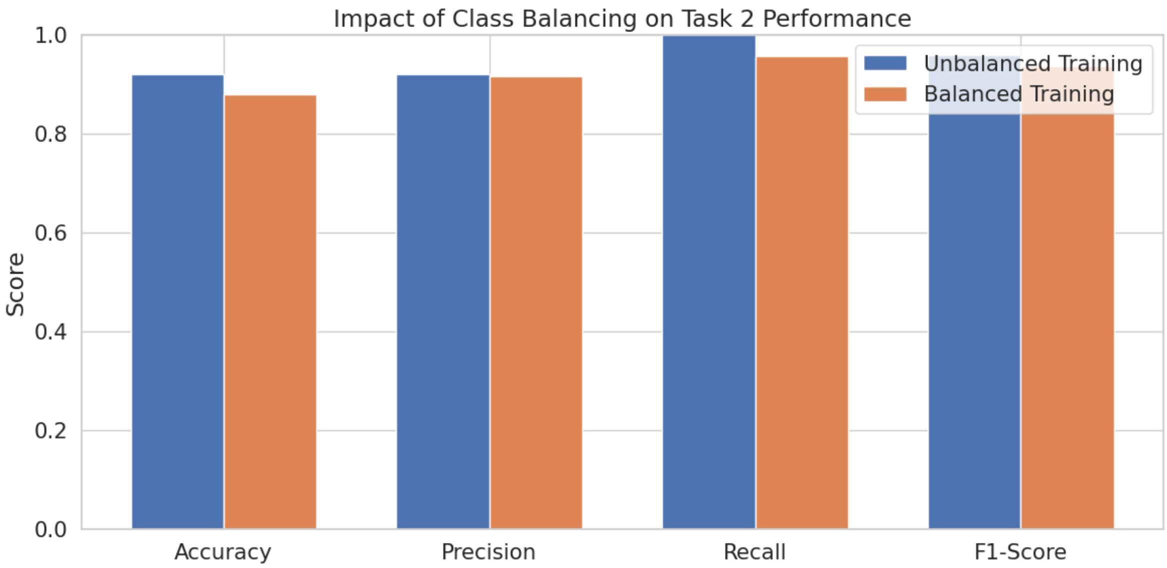


In [110…

```
# Check for class imbalance in Task 2
print("Task 2 class distribution:")
print(y_task2.value_counts())
print(y_task2.value_counts(normalize=True).round(3) * 100, '%')

# Apply class balancing for Task 2 if needed
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from sklearn.base import clone  # Add this import


# Let's try SMOTE for handling imbalance
smote = SMOTE(random_state=42)
X_train_task2_balanced, y_train_task2_balanced = smote.fit_resample(X_tra

print("\nClass distribution after SMOTE balancing:")
print(pd.Series(y_train_task2_balanced).value_counts())

# Train the best model with balanced data
best_model_balanced = clone(best_model_task2)
best_model_balanced.fit(X_train_task2_balanced, y_train_task2_balanced)

# Evaluate on the test set
y_pred_balanced = best_model_balanced.predict(X_test_task2)
accuracy_balanced = accuracy_score(y_test_task2, y_pred_balanced)
print(f"\nTest Accuracy with Balanced Training Data: {accuracy_balanced:.
print("\nClassification Report with Balanced Training Data:")
print(classification_report(y_test_task2, y_pred_balanced))
```

```python
# Compare balanced vs unbalanced training
plt.figure(figsize=(12, 6))
metrics = ['Accuracy', 'Precision', 'Recall', 'F1-Score']
unbalanced_scores = [
    accuracy_task2,
    precision_score(y_test_task2, y_pred_task2),
    recall_score(y_test_task2, y_pred_task2),
    f1_score(y_test_task2, y_pred_task2)
]
balanced_scores = [
    accuracy_balanced,
    precision_score(y_test_task2, y_pred_balanced),
    recall_score(y_test_task2, y_pred_balanced),
    f1_score(y_test_task2, y_pred_balanced)
]

x = np.arange(len(metrics))
width = 0.35

plt.bar(x - width/2, unbalanced_scores, width, label='Unbalanced Training
plt.bar(x + width/2, balanced_scores, width, label='Balanced Training')
plt.xticks(x, metrics)
plt.ylabel('Score')
plt.title('Impact of Class Balancing on Task 2 Performance')
plt.legend()
plt.ylim(0, 1)
plt.tight_layout()
plt.show()
```

```
Task 2 class distribution:
task2_label
1    342
0     33
Name: count, dtype: int64
task2_label
1    91.2
0     8.8
Name: proportion, dtype: float64 %

Class distribution after SMOTE balancing:
task2_label
1    274
0    274
Name: count, dtype: int64

Test Accuracy with Balanced Training Data: 0.8800

Classification Report with Balanced Training Data:
              precision    recall  f1-score   support

           0       0.25      0.14      0.18         7
           1       0.92      0.96      0.94        68

    accuracy                           0.88        75
   macro avg       0.58      0.55      0.56        75
weighted avg       0.85      0.88      0.86        75
```



In [111…

```
# TASK 3: Scientific Type Classification
print("Task 3: Scientific Type Classification\n" + "="*50)

# We already have the preprocessed features for science-related tweets (X
# And we have the labels in df_sci['task3_label'] (with values 0, 1, 2 fo
y_task3 = df_sci['task3_label']

# Split data for Task 3
X_train_task3, X_test_task3, y_train_task3, y_test_task3 = train_test_spl
    X_dense_sci, y_task3, test_size=0.2, random_state=42
)

# Use the same models as in Task 2
task3_models = {
    'Gaussian NB': GaussianNB(var_smoothing=0.0152),
    'Multinomial NB': MultinomialNB(alpha=0.4641, fit_prior=True),
    'Complement NB': ComplementNB(alpha=2.8480, fit_prior=True),
    'Bernoulli NB': BernoulliNB(alpha=0.17475, fit_prior=True),
    'KNN': KNeighborsClassifier(n_neighbors=7, weights='uniform', metric=
    'SVM': SVC(C=1.4174742, kernel='rbf', gamma='scale', probability=True
    'Logistic Regression': LogisticRegression(C=np.float64(4.328761281083
    'Random Forest': RandomForestClassifier(n_estimators=200, max_depth=N
}

# Train and evaluate models for Task 3
task3_results = {}
for name, model in task3_models.items():
    task3_results[name] = evaluate_model(
        model, X_train_task3, X_test_task3, y_train_task3, y_test_task3,
    )
    print("\n")
```

```
Task 3: Scientific Type Classification
==================================================
Model: Gaussian NB
Accuracy: 0.7600
              precision    recall  f1-score   support

           0       0.86      0.89      0.87        54
           1       0.67      0.43      0.52        14
           2       0.30      0.43      0.35         7

    accuracy                           0.76        75
   macro avg       0.61      0.58      0.58        75
weighted avg       0.77      0.76      0.76        75



Model: Multinomial NB
Accuracy: 0.7600
              precision    recall  f1-score   support

           0       0.76      1.00      0.86        54
           1       0.75      0.21      0.33        14
           2       0.00      0.00      0.00         7

    accuracy                           0.76        75
   macro avg       0.50      0.40      0.40        75
weighted avg       0.69      0.76      0.68        75



Model: Complement NB
Accuracy: 0.6000
              precision    recall  f1-score   support

           0       0.78      0.72      0.75        54
           1       0.31      0.29      0.30        14
           2       0.17      0.29      0.21         7

    accuracy                           0.60        75
   macro avg       0.42      0.43      0.42        75
weighted avg       0.63      0.60      0.61        75



Model: Bernoulli NB
Accuracy: 0.7067
              precision    recall  f1-score   support

           0       0.78      0.91      0.84        54
           1       0.43      0.21      0.29        14
           2       0.20      0.14      0.17         7

    accuracy                           0.71        75
   macro avg       0.47      0.42      0.43        75
weighted avg       0.66      0.71      0.67        75



Model: KNN
Accuracy: 0.7200
```

```
              precision    recall  f1-score   support

           0       0.73      0.96      0.83        54
           1       0.50      0.14      0.22        14
           2       0.00      0.00      0.00         7

    accuracy                           0.72        75
   macro avg       0.41      0.37      0.35        75
weighted avg       0.62      0.72      0.64        75


Model: SVM
Accuracy: 0.7733
              precision    recall  f1-score   support

           0       0.79      1.00      0.89        54
           1       0.60      0.21      0.32        14
           2       0.50      0.14      0.22         7

    accuracy                           0.77        75
   macro avg       0.63      0.45      0.47        75
weighted avg       0.73      0.77      0.72        75
```

```
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/hurel/Documents/repo/projet-ml/.venv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 in labels with no predicted samples. Us
e `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
Model: Logistic Regression
Accuracy: 0.7067
              precision    recall  f1-score   support

           0       0.77      0.89      0.83        54
           1       0.33      0.29      0.31        14
           2       1.00      0.14      0.25         7

    accuracy                           0.71        75
   macro avg       0.70      0.44      0.46        75
weighted avg       0.71      0.71      0.68        75



Model: Random Forest
Accuracy: 0.6667
              precision    recall  f1-score   support

           0       0.82      0.78      0.80        54
           1       0.29      0.43      0.34        14
           2       0.67      0.29      0.40         7

    accuracy                           0.67        75
   macro avg       0.59      0.50      0.51        75
weighted avg       0.71      0.67      0.68        75
```

In [112…

```
# Cross-validation for Task 3
print("Task 3: Cross-Validation\n" + "="*40)

# Initialize k-fold cross validation
k_folds = 10
skf = StratifiedKFold(n_splits=k_folds, shuffle=True, random_state=42)

# Dictionary to store CV results
task3_cv_results = {}

# Perform k-fold cross validation for each model
for model_name, model in task3_models.items():
    print(f"Performing {k_folds}-fold cross-validation for {model_name}..

    # Initialize lists to store performance metrics for each fold
    fold_accuracy = []
    fold_precision = []
    fold_recall = []
    fold_f1 = []

    # For each fold
    for fold, (train_idx, test_idx) in enumerate(skf.split(X_dense_sci, y
        # Split data
        X_train_fold, X_test_fold = X_dense_sci[train_idx], X_dense_sci[t
        y_train_fold, y_test_fold = y_task3.iloc[train_idx], y_task3.iloc

        # Train model
        model.fit(X_train_fold, y_train_fold)

        # Make predictions
        y_pred_fold = model.predict(X_test_fold)
```

```
        # Calculate metrics
        acc = accuracy_score(y_test_fold, y_pred_fold)
        # For multi-class classification, use macro averaging
        prec = precision_score(y_test_fold, y_pred_fold, average='macro',
        rec = recall_score(y_test_fold, y_pred_fold, average='macro', zer
        f1 = f1_score(y_test_fold, y_pred_fold, average='macro', zero_div

        fold_accuracy.append(acc)
        fold_precision.append(prec)
        fold_recall.append(rec)
        fold_f1.append(f1)

    # Store average metrics and standard deviations
    task3_cv_results[model_name] = {
        'accuracy': {
            'mean': np.mean(fold_accuracy),
            'std': np.std(fold_accuracy)
        },
        'precision': {
            'mean': np.mean(fold_precision),
            'std': np.std(fold_precision)
        },
        'recall': {
            'mean': np.mean(fold_recall),
            'std': np.std(fold_recall)
        },
        'f1': {
            'mean': np.mean(fold_f1),
            'std': np.std(fold_f1)
        }
    }

    print(f"  Average: Accuracy={task3_cv_results[model_name]['accuracy']
    print(f"  Average: F1 Score={task3_cv_results[model_name]['f1']['mean
    print()
```

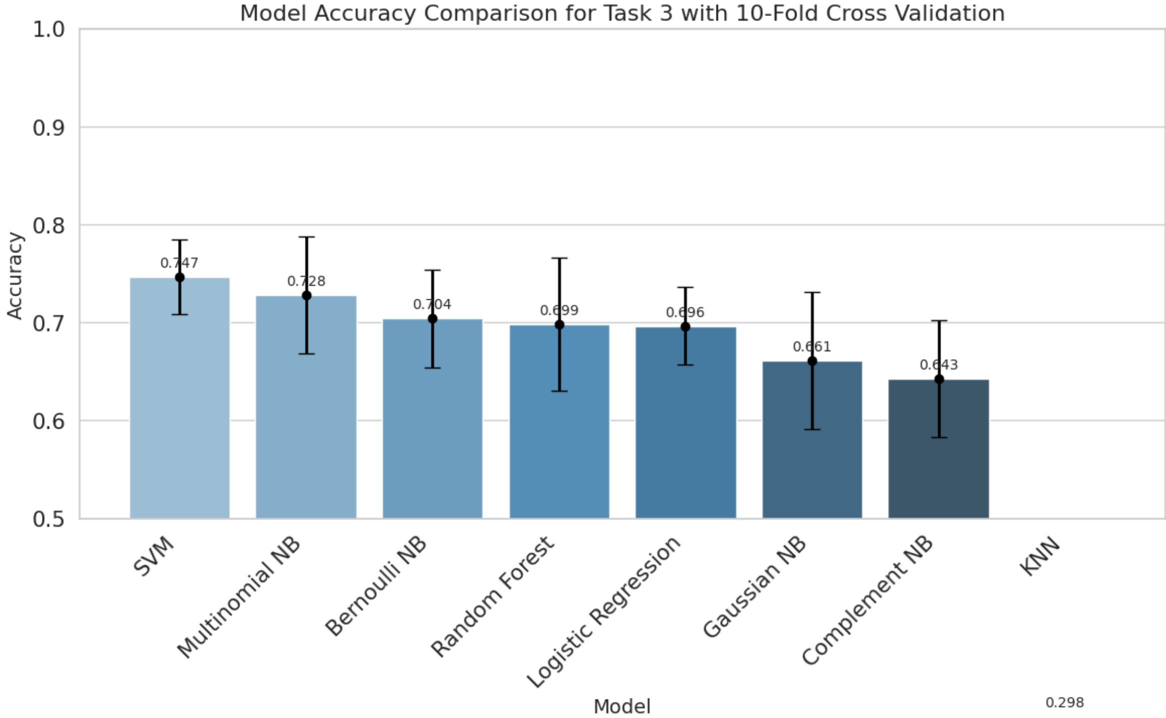
```
Task 3: Cross-Validation
========================================
Performing 10-fold cross-validation for Gaussian NB...
  Average: Accuracy=0.6612 ± 0.0701
  Average: F1 Score=0.4518 ± 0.0643

Performing 10-fold cross-validation for Multinomial NB...
  Average: Accuracy=0.7280 ± 0.0599
  Average: F1 Score=0.4004 ± 0.0922

Performing 10-fold cross-validation for Complement NB...
  Average: Accuracy=0.6428 ± 0.0595
  Average: F1 Score=0.4886 ± 0.0616

Performing 10-fold cross-validation for Bernoulli NB...
  Average: Accuracy=0.7042 ± 0.0498
  Average: F1 Score=0.4997 ± 0.0956

Performing 10-fold cross-validation for KNN...
  Average: Accuracy=0.2983 ± 0.0671
  Average: F1 Score=0.1982 ± 0.0542

Performing 10-fold cross-validation for SVM...
  Average: Accuracy=0.7467 ± 0.0379
  Average: F1 Score=0.4304 ± 0.0634

Performing 10-fold cross-validation for Logistic Regression...
  Average: Accuracy=0.6962 ± 0.0395
  Average: F1 Score=0.4445 ± 0.0834

Performing 10-fold cross-validation for Random Forest...
  Average: Accuracy=0.6985 ± 0.0678
  Average: F1 Score=0.4920 ± 0.1049
```

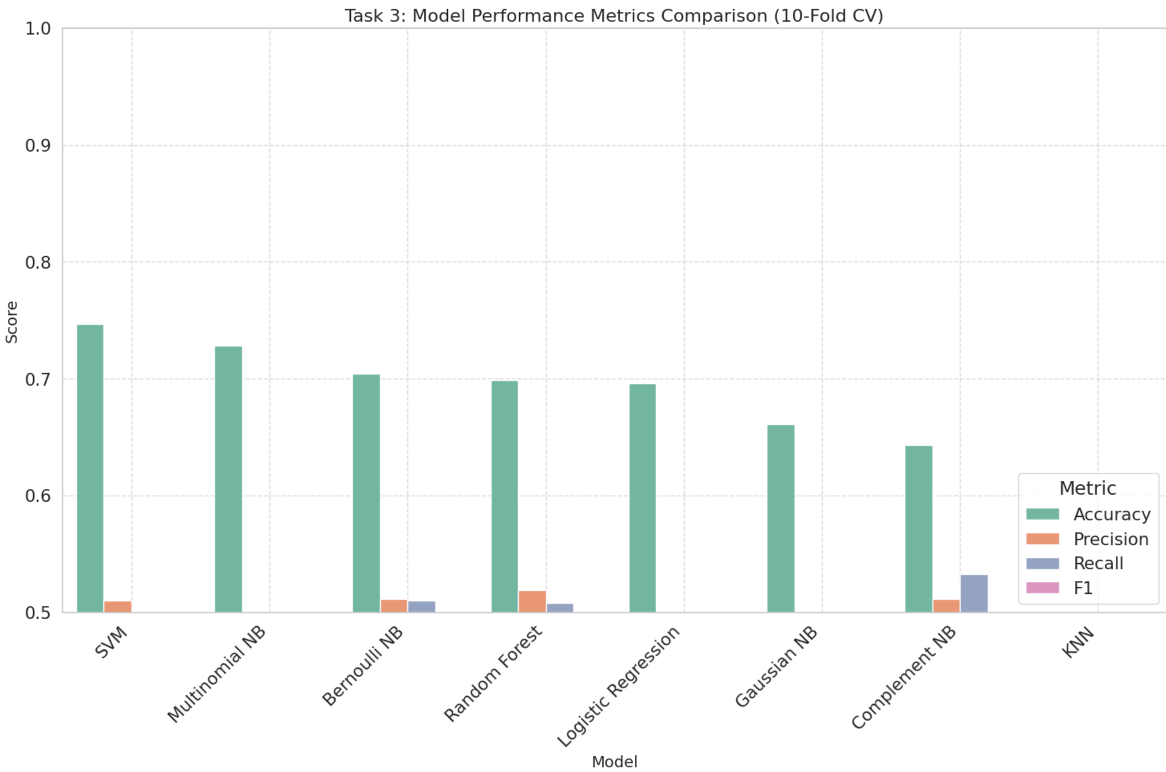
In [113…

```python
# Visualize Task 3 cross-validation results
# Create DataFrame for visualization
task3_results_df = pd.DataFrame({
    'Model': [],
    'Metric': [],
    'Mean': [],
    'Std': []
})

for model_name in task3_cv_results:
    for metric in ['accuracy', 'precision', 'recall', 'f1']:
        task3_results_df = pd.concat([task3_results_df, pd.DataFrame({
            'Model': [model_name],
            'Metric': [metric.capitalize()],
            'Mean': [task3_cv_results[model_name][metric]['mean']],
            'Std': [task3_cv_results[model_name][metric]['std']]
        })], ignore_index=True)

# Sort models by accuracy
task3_model_order = task3_results_df[task3_results_df['Metric'] == 'Accur

# Create accuracy bar plot
plt.figure(figsize=(12, 8))
sns.set_style("whitegrid")
```

```
# Create bar plot for accuracy
ax = sns.barplot(
    data=task3_results_df[task3_results_df['Metric'] == 'Accuracy'],
    x='Model',
    y='Mean',
    order=task3_model_order,
    palette='Blues_d'
)

# Add error bars
for i, model in enumerate(task3_model_order):
    row = task3_results_df[(task3_results_df['Model'] == model) & (task3_
    ax.errorbar(
        i, row['Mean'], yerr=row['Std'],
        fmt='o', color='black', elinewidth=2, capsize=6
    )

# Add value labels on top of bars
for i, bar in enumerate(ax.patches):
    ax.text(
        bar.get_x() + bar.get_width()/2,
        bar.get_height() + 0.01,
        f"{bar.get_height():.3f}",
        ha='center',
        fontsize=10
    )

plt.title(f'Model Accuracy Comparison for Task 3 with {k_folds}-Fold Cros
plt.xlabel('Model', fontsize=14)
plt.ylabel('Accuracy', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.ylim([0.5, 1.0])
plt.tight_layout()
plt.show()

# Create a grouped bar chart for all metrics
plt.figure(figsize=(15, 10))
sns.set_style("whitegrid")

# Create grouped bar plot
ax = sns.barplot(
    data=task3_results_df,
    x='Model',
    y='Mean',
    hue='Metric',
    order=task3_model_order,
    palette='Set2'
)

plt.title(f'Task 3: Model Performance Metrics Comparison ({k_folds}-Fold
plt.xlabel('Model', fontsize=14)
plt.ylabel('Score', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.legend(title='Metric', loc='lower right')
plt.ylim([0.5, 1.0])
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
/tmp/ipykernel_368067/1263538886.py:27: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.

  ax = sns.barplot(
```





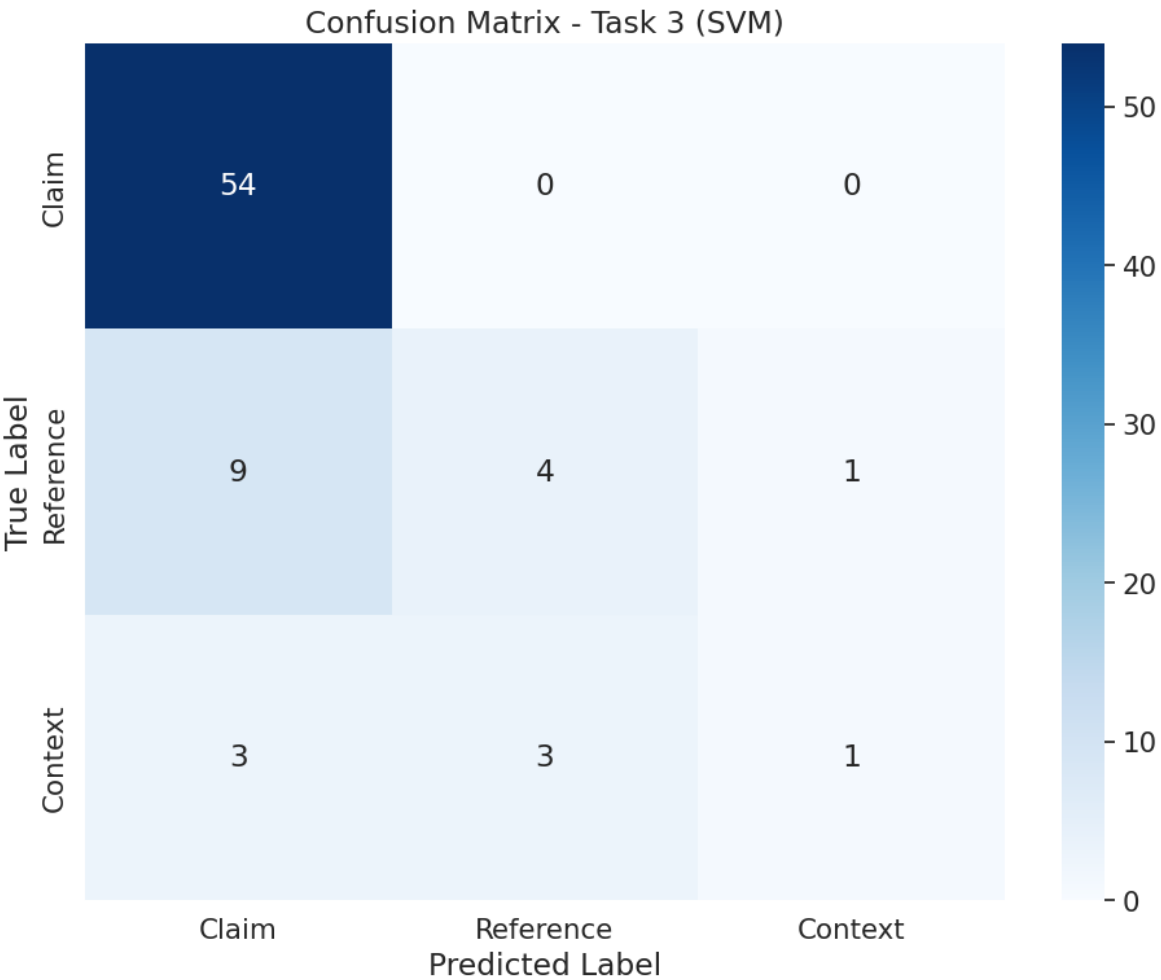
In [114…

```
# Choose the best model from cross-validation for Task 3
best_task3_model_name = task3_model_order[0]  # The model with highest ac
best_task3_model = task3_models[best_task3_model_name]

# Hyperparameter optimization for the best model on Task 3
print(f"Performing Grid Search for {best_task3_model_name} on Task 3...")
```

```python
# Define the parameter grid for the best model
if best_task3_model_name == 'SVM':
    param_grid = {
        'C': np.logspace(-3, 3, 7),
        'gamma': ['scale', 'auto'] + list(np.logspace(-3, 3, 5)),
        'kernel': ['rbf', 'linear']
    }
elif best_task3_model_name == 'Random Forest':
    param_grid = {
        'n_estimators': [50, 100, 200],
        'max_depth': [None, 10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }
elif best_task3_model_name.endswith('NB'):
    if best_task3_model_name == 'Gaussian NB':
        param_grid = {
            'var_smoothing': np.logspace(-10, 0, 11)
        }
    else:
        param_grid = {
            'alpha': np.logspace(-3, 3, 7),
            'fit_prior': [True, False]
        }
else:
    # Default grid for other models
    param_grid = {
        'C': np.logspace(-3, 3, 7)
    }

# Create and fit GridSearchCV
grid_search_task3 = GridSearchCV(
    estimator=best_task3_model,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=1
)

grid_search_task3.fit(X_train_task3, y_train_task3)

# Display the best parameters and score
print(f"Best Parameters: {grid_search_task3.best_params_}")
print(f"Best Cross-Validation Score: {grid_search_task3.best_score_:.4f}"

# Evaluate on test set
best_model_task3 = grid_search_task3.best_estimator_
y_pred_task3 = best_model_task3.predict(X_test_task3)
accuracy_task3 = accuracy_score(y_test_task3, y_pred_task3)
print(f"Test Accuracy with Best Parameters: {accuracy_task3:.4f}")
print("\nClassification Report:")
print(classification_report(y_test_task3, y_pred_task3))

# Plot confusion matrix
plt.figure(figsize=(10, 8))
cm = confusion_matrix(y_test_task3, y_pred_task3)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Claim', 'Reference', 'Context'],
            yticklabels=['Claim', 'Reference', 'Context'])
```

```
plt.title(f'Confusion Matrix - Task 3 ({best_task3_model_name})')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.tight_layout()
plt.show()
```

```
Performing Grid Search for SVM on Task 3...
Fitting 5 folds for each of 98 candidates, totalling 490 fits
Best Parameters: {'C': np.float64(10.0), 'gamma': 'scale', 'kernel': 'rb
f'}
Best Cross-Validation Score: 0.7433
Test Accuracy with Best Parameters: 0.7867

Classification Report:
              precision    recall  f1-score   support

           0       0.82      1.00      0.90        54
           1       0.57      0.29      0.38        14
           2       0.50      0.14      0.22         7

    accuracy                           0.79        75
   macro avg       0.63      0.48      0.50        75
weighted avg       0.74      0.79      0.74        75
```

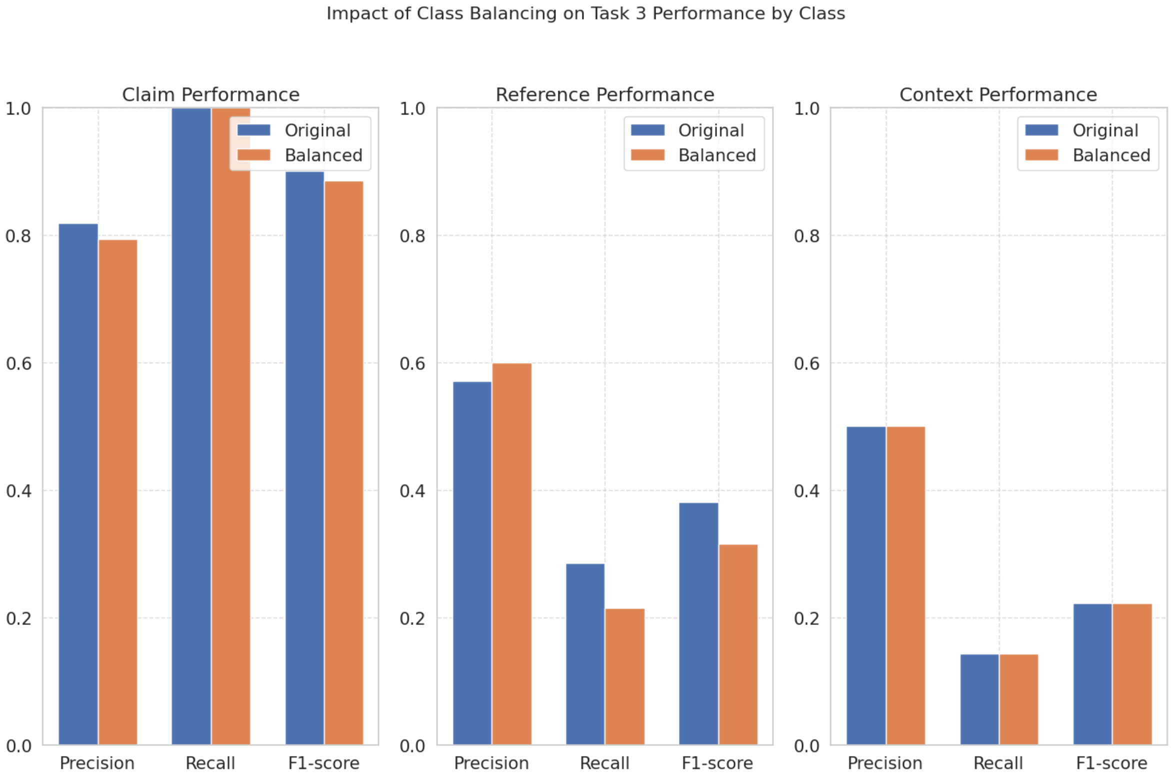


In [115…

```
# Check for class imbalance in Task 3
print("Task 3 class distribution:")
print(y_task3.value_counts())
print(y_task3.value_counts(normalize=True).round(3) * 100, '%')

# Apply SMOTE for handling class imbalance in Task 3
```

```
smote = SMOTE(random_state=42)
X_train_task3_balanced, y_train_task3_balanced = smote.fit_resample(X_tra

print("\nClass distribution after SMOTE balancing:")
print(pd.Series(y_train_task3_balanced).value_counts())

# Train the best model with balanced data
best_model_balanced = clone(best_model_task3)
best_model_balanced.fit(X_train_task3_balanced, y_train_task3_balanced)

# Evaluate on the test set
y_pred_balanced = best_model_balanced.predict(X_test_task3)
accuracy_balanced = accuracy_score(y_test_task3, y_pred_balanced)
print(f"\nTest Accuracy with Balanced Training Data: {accuracy_balanced:.
print("\nClassification Report with Balanced Training Data:")
print(classification_report(y_test_task3, y_pred_balanced))

# Compare class-wise performance before and after balancing
original_report = classification_report(y_test_task3, y_pred_task3, outpu
balanced_report = classification_report(y_test_task3, y_pred_balanced, ou

# Create a DataFrame to compare per-class metrics
class_comparison = []
for class_label in ['0', '1', '2']:  # Claim, Reference, Context
    for metric in ['precision', 'recall', 'f1-score']:
        class_comparison.append({
            'Class': ['Claim', 'Reference', 'Context'][int(class_label)],
            'Metric': metric.capitalize(),
            'Original': original_report[class_label][metric],
            'Balanced': balanced_report[class_label][metric],
            'Difference': balanced_report[class_label][metric] - original
        })

class_comp_df = pd.DataFrame(class_comparison)

# Plot class-wise performance comparison
plt.figure(figsize=(15, 10))

for i, cls in enumerate(['Claim', 'Reference', 'Context']):
    plt.subplot(1, 3, i+1)
    df_class = class_comp_df[class_comp_df['Class'] == cls]

    x = np.arange(len(df_class['Metric'].unique()))
    width = 0.35

    orig_scores = df_class['Original'].values
    bal_scores = df_class['Balanced'].values

    plt.bar(x - width/2, orig_scores, width, label='Original')
    plt.bar(x + width/2, bal_scores, width, label='Balanced')

    plt.title(f'{cls} Performance')
    plt.xticks(x, df_class['Metric'].unique())
    plt.ylim(0, 1)
    plt.legend()
    plt.grid(True, linestyle='--', alpha=0.7)

plt.suptitle('Impact of Class Balancing on Task 3 Performance by Class',
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

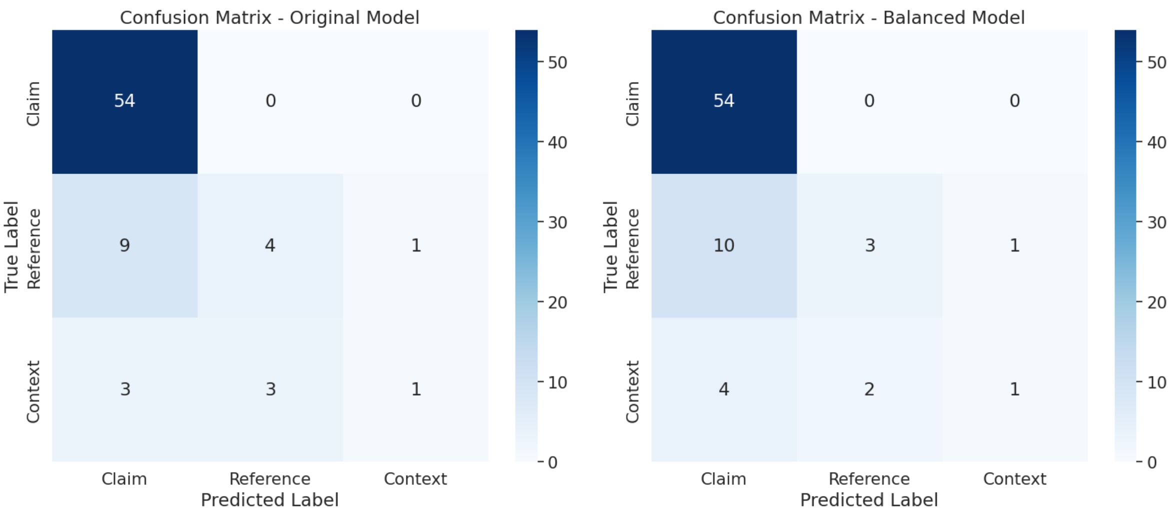
```
Task 3 class distribution:
task3_label
0    263
1     79
2     33
Name: count, dtype: int64
task3_label
0    70.1
1    21.1
2     8.8
Name: proportion, dtype: float64 %

Class distribution after SMOTE balancing:
task3_label
0    209
1    209
2    209
Name: count, dtype: int64

Test Accuracy with Balanced Training Data: 0.7733

Classification Report with Balanced Training Data:
              precision    recall  f1-score   support

           0       0.79      1.00      0.89        54
           1       0.60      0.21      0.32        14
           2       0.50      0.14      0.22         7

    accuracy                           0.77        75
   macro avg       0.63      0.45      0.47        75
weighted avg       0.73      0.77      0.72        75
```



In [116…

```
# Visualize confusion matrices for balanced vs unbalanced
plt.figure(figsize=(16, 7))

# Original model confusion matrix
```

```python
plt.subplot(1, 2, 1)
cm_orig = confusion_matrix(y_test_task3, y_pred_task3)
sns.heatmap(cm_orig, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Claim', 'Reference', 'Context'],
            yticklabels=['Claim', 'Reference', 'Context'])
plt.title('Confusion Matrix - Original Model')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')

# Balanced model confusion matrix
plt.subplot(1, 2, 2)
cm_balanced = confusion_matrix(y_test_task3, y_pred_balanced)
sns.heatmap(cm_balanced, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Claim', 'Reference', 'Context'],
            yticklabels=['Claim', 'Reference', 'Context'])
plt.title('Confusion Matrix - Balanced Model')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')

plt.tight_layout()
plt.show()
```



In [126…

```python
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB, MultinomialNB, ComplementNB,
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import emoji
from sklearn.base import BaseEstimator, TransformerMixin
from scipy.sparse import csr_matrix

# Custom text preprocessor
class TextPreprocessor(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.lemmatizer = WordNetLemmatizer()
        self.stop_words = set(stopwords.words('english'))

    def fit(self, X, y=None):
```

```
        return self

    def transform(self, X):
        return [self.preprocess(text) for text in X]

    def preprocess(self, text):
        # Convert to lowercase
        text = text.lower()

        # Demojize text
        text = emoji.demojize(text)

        # Remove URLs
        text = re.sub(r'(http\S+|www\S+)', '', text)

        # Remove mentions and hashtags except for the word eurekamag
        text = re.sub(r'@\w+', '', text)
        text = re.sub(r'#eurekamag', 'eurekamag', text)
        text = re.sub(r'#\w+', '', text)

        # Remove special characters and numbers
        text = re.sub(r'[^a-zA-Z\s]', '', text)

        # Tokenize
        tokens = nltk.word_tokenize(text)

        # Remove stopwords
        tokens = [word for word in tokens if word not in self.stop_words]

        # Lemmatize
        tokens = [self.lemmatizer.lemmatize(word) for word in tokens]

        # Rejoin
        return ' '.join(tokens)

# Convert sparse features to dense for models that need it
class DenseTransformer(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        if isinstance(X, csr_matrix):
            return X.toarray()
        return X

# Create model pipeline based on model type
def create_model_pipeline(model_name, task=1):
    # Define vectorizer
    vectorizer = TfidfVectorizer(
        max_features=5000,
        min_df=5,
        max_df=0.8,
        ngram_range=(1, 2)
    )

    # Create model with optimized parameters
    if model_name == 'Gaussian NB':
        model = GaussianNB(var_smoothing=0.0152)
        return Pipeline([
            ('preprocessor', TextPreprocessor()),
```

```
            ('vectorizer', vectorizer),
            ('to_dense', DenseTransformer()),
            ('classifier', model)
        ])
    elif model_name == 'Multinomial NB':
        model = MultinomialNB(alpha=0.4641, fit_prior=True)
        return Pipeline([
            ('preprocessor', TextPreprocessor()),
            ('vectorizer', vectorizer),
            ('classifier', model)
        ])
    elif model_name == 'Complement NB':
        model = ComplementNB(alpha=2.8480, fit_prior=True)
        return Pipeline([
            ('preprocessor', TextPreprocessor()),
            ('vectorizer', vectorizer),
            ('classifier', model)
        ])
    elif model_name == 'Bernoulli NB':
        model = BernoulliNB(alpha=0.17475, fit_prior=True)
        return Pipeline([
            ('preprocessor', TextPreprocessor()),
            ('vectorizer', vectorizer),
            ('classifier', model)
        ])
    elif model_name == 'KNN':
        model = KNeighborsClassifier(n_neighbors=7, weights='uniform', me
        return Pipeline([
            ('preprocessor', TextPreprocessor()),
            ('vectorizer', vectorizer),
            ('to_dense', DenseTransformer()),
            ('classifier', model)
        ])
    elif model_name == 'SVM':
        model = SVC(C=1.4174742, kernel='rbf', gamma='scale', probability
        return Pipeline([
            ('preprocessor', TextPreprocessor()),
            ('vectorizer', vectorizer),
            ('classifier', model)
        ])
    elif model_name == 'Logistic Regression':
        model = LogisticRegression(C=4.328761281083062, penalty='l1', sol
        return Pipeline([
            ('preprocessor', TextPreprocessor()),
            ('vectorizer', vectorizer),
            ('classifier', model)
        ])
    elif model_name == 'Random Forest':
        model = RandomForestClassifier(n_estimators=200, max_depth=None,
        return Pipeline([
            ('preprocessor', TextPreprocessor()),
            ('vectorizer', vectorizer),
            ('classifier', model)
        ])
    else:
        raise ValueError(f"Unknown model: {model_name}")

# Function to create a cascading predictor for all tasks
def create_science_tweet_classifier():
    # Create individual task pipelines with best models
```

```
    task1_pipeline = create_model_pipeline('SVM', task=1)  # Science rela
    task2_pipeline = create_model_pipeline('SVM', task=2)  # Claim/refere
    task3_pipeline = create_model_pipeline('SVM', task=3)  # Type classif

    # Define prediction function
    def predict(text):
        # First determine if science related
        is_science = task1_pipeline.predict([text])[0]

        if is_science == 1:
            # If science-related, check for claim/reference
            has_claim_ref = task2_pipeline.predict([text])[0]

            # Determine science type
            sci_type_code = task3_pipeline.predict([text])[0]
            sci_type = ['Scientific Claim', 'Scientific Reference', 'Scie

            return {
                'science_related': True,
                'has_claim_or_reference': bool(has_claim_ref),
                'science_type': sci_type
            }
        else:
            return {
                'science_related': False,
                'has_claim_or_reference': None,
                'science_type': None
            }

    # Return the pipelines and prediction function
    return {
        'task1_pipeline': task1_pipeline,
        'task2_pipeline': task2_pipeline,
        'task3_pipeline': task3_pipeline,
        'predict': predict
    }
```

In [ ]: