```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression


import os

# Define the path to your folder
folder_path = '/content/drive/My Drive/Crime data'

# List all files in the folder to verify the folder path
files = os.listdir(folder_path)
print("Files in the 'crime data' folder:", files)
```

```
Files in the 'crime data' folder: ['2024-01-cleveland-street.csv', '2024-01-bedfordshire-street.csv', '2024-01-cambridgeshire-street.csv', '2024-01-avon-and-somerset-street.csv', '202
```

```
# Find the West Yorkshire street file
weststreet_file = '2024-01-west-midlands-street.csv'

# Check if the Cleveland street file exists in the folder
if weststreet_file in files:
    # Load the Cleveland street file into a DataFrame
    weststreet_df = pd.read_csv(os.path.join(folder_path, weststreet_file))

    # Display the first few rows of the Cleveland DataFrame
    print("Data from weststreet File:")
    print(weststreet_df.head())  # Display the first few rows
else:
    print(f"File {weststreet_file } not found in the folder.")
```

```
Data from weststreet File:
                                             Crime ID    Month  \
0                                                 NaN  2024-01
1                                                 NaN  2024-01
2  08fcc46d224b1e4f4065b950fa59e5251d51106f3472cd...  2024-01
3  74328959382b1b658b18199c2a26b56c9f09458a0d12a8...  2024-01
4  5ea6fbc3cd6834f56ba25852736e3e4d53a70163f2a3a3...  2024-01

             Reported by          Falls within  Longitude   Latitude  \
0  West Midlands Police  West Midlands Police  -1.849790  52.590937
1  West Midlands Police  West Midlands Police  -1.840582  52.598279
2  West Midlands Police  West Midlands Police  -1.840582  52.598279
3  West Midlands Police  West Midlands Police  -1.846312  52.593702
4  West Midlands Police  West Midlands Police  -1.839093  52.597818

                       Location  LSOA code        LSOA name  \
0       On or near Walsall Road  E01009417  Birmingham 001A
1  On or near Badgers Bank Road  E01009418  Birmingham 001B
2  On or near Badgers Bank Road  E01009418  Birmingham 001B
3         On or near Hook Drive  E01009418  Birmingham 001B
4        On or near Byron Court  E01009418  Birmingham 001B

                     Crime type  \
0         Anti-social behaviour
1         Anti-social behaviour
```

```
2                Vehicle crime
3  Violence and sexual offences
4  Violence and sexual offences

                           Last outcome category  Context
0                                            NaN      NaN
1                                            NaN      NaN
2  Investigation complete; no suspect identified      NaN
3                    Unable to prosecute suspect      NaN
4                    Unable to prosecute suspect      NaN
```

```
# Drop the unnecessary columns
df_clean = weststreet_df.drop(columns=['Crime ID','Context'])


# Removing the 'E' prefix from the 'LSOA code' column
df_clean['LSOA code'] = df_clean['LSOA code'].str.replace('^E', '', regex=True)

print(df_clean)
```

```
          Month           Reported by          Falls within  Longitude  \
2      2024-01  West Midlands Police  West Midlands Police  -1.840582
3      2024-01  West Midlands Police  West Midlands Police  -1.846312
4      2024-01  West Midlands Police  West Midlands Police  -1.839093
5      2024-01  West Midlands Police  West Midlands Police  -1.845233
6      2024-01  West Midlands Police  West Midlands Police  -1.846312
...        ...                   ...                   ...        ...
27494  2024-01  West Midlands Police  West Midlands Police  -2.113522
27496  2024-01  West Midlands Police  West Midlands Police  -2.115358
27497  2024-01  West Midlands Police  West Midlands Police  -2.118019
27498  2024-01  West Midlands Police  West Midlands Police  -2.118019
27499  2024-01  West Midlands Police  West Midlands Police  -2.118019

        Latitude                     Location LSOA code           LSOA name  \
2      52.598279  On or near Badgers Bank Road  01009418     Birmingham 001B
3      52.593702         On or near Hook Drive  01009418     Birmingham 001B
4      52.597818        On or near Byron Court  01009418     Birmingham 001B
5      52.593943     On or near Harcourt Drive  01009418     Birmingham 001B
6      52.593702         On or near Hook Drive  01009418     Birmingham 001B
...          ...                           ...       ...                ...
27494  52.576576      On or near Furnace Drive  01034314  Wolverhampton 035I
27496  52.573302      On or near Pond Crescent  01034315  Wolverhampton 035J
27497  52.569505    On or near Thompson Avenue  01034315  Wolverhampton 035J
27498  52.569505    On or near Thompson Avenue  01034315  Wolverhampton 035J
27499  52.569505    On or near Thompson Avenue  01034315  Wolverhampton 035J

                        Crime type  \
2                    Vehicle crime
3      Violence and sexual offences
4      Violence and sexual offences
5      Violence and sexual offences
6      Violence and sexual offences
...                             ...
27494  Violence and sexual offences
27496                      Robbery
27497                      Robbery
27498                Vehicle crime
27499  Violence and sexual offences

                              Last outcome category
2      Investigation complete; no suspect identified
3                          Unable to prosecute suspect
```

```
4                       Unable to prosecute suspect
5                       Unable to prosecute suspect
6                       Unable to prosecute suspect
...                                             ...
27494     Action to be taken by another organisation
27496                   Unable to prosecute suspect
27497  Investigation complete; no suspect identified
27498  Investigation complete; no suspect identified
27499                     Status update unavailable

[25713 rows x 10 columns]
```

```
# Drop rows where 'Crime ID' or 'Last outcome category' are null
df_clean = df_clean.dropna(subset=[ 'Last outcome category'])

# Check the result
print(df_clean.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 25713 entries, 2 to 27499
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Month                  25713 non-null  object
 1   Reported by            25713 non-null  object
 2   Falls within           25713 non-null  object
 3   Longitude              25713 non-null  float64
 4   Latitude               25713 non-null  float64
 5   Location               25713 non-null  object
 6   LSOA code              25713 non-null  object
 7   LSOA name              25713 non-null  object
 8   Crime type             25713 non-null  object
 9   Last outcome category  25713 non-null  object
dtypes: float64(2), object(8)
memory usage: 2.2+ MB
None
```

```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.5.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
```

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
```

```
df_clean.head()
```

	Month	Reported by	Falls within	Longitude	Latitude	Location	LSOA code	LSOA name	Crime type	Last outcome category
2	2024-01	West Midlands Police	West Midlands Police	-1.840582	52.598279	On or near Badgers Bank Road	E01009418	Birmingham 001B	Vehicle crime	Investigation complete; no suspect identified
3	2024-01	West Midlands Police	West Midlands Police	-1.846312	52.593702	On or near Hook Drive	E01009418	Birmingham 001B	Violence and sexual offences	Unable to prosecute suspect
4	2024-01	West Midlands Police	West Midlands Police	-1.839093	52.597818	On or near Byron Court	E01009418	Birmingham 001B	Violence and sexual offences	Unable to prosecute suspect

```
# Replace semicolons with nothing (effectively removing them)
df_clean['Last outcome category'] = df_clean['Last outcome category'].str.replace(';', '', regex=False)

print(df_clean)
```

```
         Month           Reported by          Falls within  Longitude  \
2      2024-01  West Midlands Police  West Midlands Police  -1.840582
3      2024-01  West Midlands Police  West Midlands Police  -1.846312
4      2024-01  West Midlands Police  West Midlands Police  -1.839093
5      2024-01  West Midlands Police  West Midlands Police  -1.845233
6      2024-01  West Midlands Police  West Midlands Police  -1.846312
...        ...                   ...                   ...        ...
27494  2024-01  West Midlands Police  West Midlands Police  -2.113522
27496  2024-01  West Midlands Police  West Midlands Police  -2.115358
27497  2024-01  West Midlands Police  West Midlands Police  -2.118019
27498  2024-01  West Midlands Police  West Midlands Police  -2.118019
27499  2024-01  West Midlands Police  West Midlands Police  -2.118019

        Latitude                     Location  LSOA code           LSOA name  \
2      52.598279  On or near Badgers Bank Road  E01009418     Birmingham 001B
3      52.593702         On or near Hook Drive  E01009418     Birmingham 001B
4      52.597818        On or near Byron Court  E01009418     Birmingham 001B
5      52.593943     On or near Harcourt Drive  E01009418     Birmingham 001B
6      52.593702         On or near Hook Drive  E01009418     Birmingham 001B
...          ...                           ...        ...                ...
27494  52.576576      On or near Furnace Drive  E01034314  Wolverhampton 035I
27496  52.573302      On or near Pond Crescent  E01034315  Wolverhampton 035J
27497  52.569505    On or near Thompson Avenue  E01034315  Wolverhampton 035J
27498  52.569505    On or near Thompson Avenue  E01034315  Wolverhampton 035J
27499  52.569505    On or near Thompson Avenue  E01034315  Wolverhampton 035J

                      Crime type  \
2                   Vehicle crime
3      Violence and sexual offences
4      Violence and sexual offences
5      Violence and sexual offences
6      Violence and sexual offences
...                             ...
27494  Violence and sexual offences
27496                      Robbery
27497                      Robbery
27498                 Vehicle crime
27499  Violence and sexual offences

                           Last outcome category
2      Investigation complete no suspect identified
3                        Unable to prosecute suspect
4                        Unable to prosecute suspect
5                        Unable to prosecute suspect
```

```
6                     Unable to prosecute suspect
...                                           ...
27494    Action to be taken by another organisation
27496                   Unable to prosecute suspect
27497  Investigation complete no suspect identified
27498  Investigation complete no suspect identified
27499                     Status update unavailable

[25713 rows x 10 columns]
```

```
from sklearn.feature_extraction.text import CountVectorizer

# Define your custom list of stop words
custom_stop_words = ['no', 'to', 'by', 'is', 'not', 'in', 'the','be','as']

# Initialize the CountVectorizer with your custom stop words
vectorizer = CountVectorizer(stop_words=custom_stop_words)

# Fit and transform the data
X = vectorizer.fit_transform(df_clean['Last outcome category'])

# Convert to DataFrame for better readability
bagow_df = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out())

print(bagow_df)
```

```
       action  another  awaiting  case  caution  charged  complete  court  \
0           0        0         0     0        0        0         1      0
1           0        0         0     0        0        0         0      0
2           0        0         0     0        0        0         0      0
3           0        0         0     0        0        0         0      0
4           0        0         0     0        0        0         0      0
...       ...      ...       ...   ...      ...      ...       ...    ...
25708       1        1         0     0        0        0         0      0
25709       0        0         0     0        0        0         0      0
25710       0        0         0     0        0        0         1      0
25711       0        0         0     0        0        0         1      0
25712       0        0         0     0        0        0         0      0

       formal  further  ...  part  prosecute  public  resolution  status  \
0           0        0  ...     0          0       0           0       0
1           0        0  ...     0          1       0           0       0
2           0        0  ...     0          1       0           0       0
3           0        0  ...     0          1       0           0       0
4           0        0  ...     0          1       0           0       0
...       ...      ...  ...   ...        ...     ...         ...     ...
25708       0        0  ...     0          0       0           0       0
25709       0        0  ...     0          1       0           0       0
25710       0        0  ...     0          0       0           0       0
25711       0        0  ...     0          0       0           0       0
25712       0        0  ...     0          0       0           0       1

       suspect  taken  unable  unavailable  update
0            1      0       0            0       0
1            1      0       1            0       0
2            1      0       1            0       0
3            1      0       1            0       0
4            1      0       1            0       0
...        ...    ...     ...          ...     ...
25708        0      1       0            0       0
```

```
25709        1      0       1           0       0
25710        1      0       0           0       0
25711        1      0       0           0       0
25712        0      0       0           1       1

[25713 rows x 29 columns]
```

```
from sklearn.cluster import KMeans

# Define the number of clusters
n_clusters = 10

# Explicitly set the `n_init` parameter
kmeans = KMeans(n_clusters=n_clusters, n_init=10)  # You can set this to 10 or any other number you prefer

# Fit the model
df_clean['Zone'] = kmeans.fit_predict(df_clean[['Longitude', 'Latitude']].dropna())
```

```
df_clean
```

	Month	Reported by	Falls within	Longitude	Latitude	Location	LSOA code	LSOA name	Crime type	Last outcome category	Zone
2	2024-01	West Midlands Police	West Midlands Police	-1.840582	52.598279	On or near Badgers Bank Road	E01009418	Birmingham 001B	Vehicle crime	Investigation complete no suspect identified	9
3	2024-01	West Midlands Police	West Midlands Police	-1.846312	52.593702	On or near Hook Drive	E01009418	Birmingham 001B	Violence and sexual offences	Unable to prosecute suspect	9
4	2024-01	West Midlands Police	West Midlands Police	-1.839093	52.597818	On or near Byron Court	E01009418	Birmingham 001B	Violence and sexual offences	Unable to prosecute suspect	9
5	2024-01	West Midlands Police	West Midlands Police	-1.845233	52.593943	On or near Harcourt Drive	E01009418	Birmingham 001B	Violence and sexual offences	Unable to prosecute suspect	9
6	2024-01	West Midlands Police	West Midlands Police	-1.846312	52.593702	On or near Hook Drive	E01009418	Birmingham 001B	Violence and sexual offences	Unable to prosecute suspect	9
...	...	...	...	...	...	...	...	...	...	...	...
27494	2024-01	West Midlands Police	West Midlands Police	-2.113522	52.576576	On or near Furnace Drive	E01034314	Wolverhampton 035I	Violence and sexual offences	Action to be taken by another organisation	8
27496	2024-01	West Midlands Police	West Midlands Police	-2.115358	52.573302	On or near Pond Crescent	E01034315	Wolverhampton 035J	Robbery	Unable to prosecute suspect	8
27497	2024-01	West Midlands Police	West Midlands Police	-2.118019	52.569505	On or near Thompson Avenue	E01034315	Wolverhampton 035J	Robbery	Investigation complete no suspect identified	8
27498	2024-01	West Midlands Police	West Midlands Police	-2.118019	52.569505	On or near Thompson Avenue	E01034315	Wolverhampton 035J	Vehicle crime	Investigation complete no suspect identified	8
27499	2024-01	West Midlands Police	West Midlands Police	-2.118019	52.569505	On or near Thompson Avenue	E01034315	Wolverhampton 035J	Violence and sexual offences	Status update unavailable	8

25713 rows × 11 columns

```
# Drop the specified columns
columns_to_drop = ['Reported by','Longitude','Latitude','Falls within','Month']
df_clean = df_clean.drop(columns=columns_to_drop)


df_clean
```

	Location	LSOA code	LSOA name	Crime type	Last outcome category	Zone
2	On or near Badgers Bank Road	E01009418	Birmingham 001B	Vehicle crime	Investigation complete no suspect identified	7
3	On or near Hook Drive	E01009418	Birmingham 001B	Violence and sexual offences	Unable to prosecute suspect	7
4	On or near Byron Court	E01009418	Birmingham 001B	Violence and sexual offences	Unable to prosecute suspect	7
5	On or near Harcourt Drive	E01009418	Birmingham 001B	Violence and sexual offences	Unable to prosecute suspect	7
6	On or near Hook Drive	E01009418	Birmingham 001B	Violence and sexual offences	Unable to prosecute suspect	7
...	...	...	...	...	...	...
27494	On or near Furnace Drive	E01034314	Wolverhampton 035I	Violence and sexual offences	Action to be taken by another organisation	3
27496	On or near Pond Crescent	E01034315	Wolverhampton 035J	Robbery	Unable to prosecute suspect	3
27497	On or near Thompson Avenue	E01034315	Wolverhampton 035J	Robbery	Investigation complete no suspect identified	3
27498	On or near Thompson Avenue	E01034315	Wolverhampton 035J	Vehicle crime	Investigation complete no suspect identified	3
27499	On or near Thompson Avenue	E01034315	Wolverhampton 035J	Violence and sexual offences	Status update unavailable	3

25713 rows × 6 columns

```
# Importing regular expression module
import re

# Function to remove unwanted phrases
def clean_location(text):
    # Use regex to remove phrases like "On or near"
    return re.sub(r'^On or near\s+', '', text)

# Apply the cleaning function to the Location column
df_clean['Location'] = df_clean['Location'].apply(clean_location)

# Display the cleaned DataFrame
print(df_clean)
```

```
                Location LSOA code           LSOA name  \
2      Badgers Bank Road  01009418     Birmingham 001B
3             Hook Drive  01009418     Birmingham 001B
4           Byron Court  01009418     Birmingham 001B
5         Harcourt Drive  01009418     Birmingham 001B
6             Hook Drive  01009418     Birmingham 001B
...                  ...       ...                 ...
27494      Furnace Drive  01034314  Wolverhampton 035I
27496      Pond Crescent  01034315  Wolverhampton 035J
27497    Thompson Avenue  01034315  Wolverhampton 035J
27498    Thompson Avenue  01034315  Wolverhampton 035J
27499    Thompson Avenue  01034315  Wolverhampton 035J

                      Crime type  \
2                   Vehicle crime
3      Violence and sexual offences
```

```
4      Violence and sexual offences
5      Violence and sexual offences
6      Violence and sexual offences
...                             ...
27494  Violence and sexual offences
27496                       Robbery
27497                       Robbery
27498                 Vehicle crime
27499  Violence and sexual offences

                            Last outcome category  Zone
2      Investigation complete; no suspect identified     9
3                      Unable to prosecute suspect     9
4                      Unable to prosecute suspect     9
5                      Unable to prosecute suspect     9
6                      Unable to prosecute suspect     9
...                                            ...   ...
27494     Action to be taken by another organisation     7
27496                  Unable to prosecute suspect     7
27497  Investigation complete; no suspect identified     7
27498  Investigation complete; no suspect identified     7
27499                    Status update unavailable     7

[25713 rows x 6 columns]
```

```
unique_lsoa_codes
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-e700ca23f9de> in <cell line: 1>()
----> 1 unique_lsoa_codes

NameError: name 'unique_lsoa_codes' is not defined
```

```
# Get unique LSOA codes
unique_lsoa_codes = df_clean['LSOA code'].unique()

# Get unique LSOA names
unique_lsoa_names = df_clean['LSOA name'].unique()

# Print the unique values
print("Unique LSOA Codes:", unique_lsoa_codes)
print("Unique LSOA Names:", unique_lsoa_names)
```

```
Unique LSOA Codes: ['01009418' '01009419' '01009433' ... '01010530' '01034314' '01034315']
Unique LSOA Names: ['Birmingham 001B' 'Birmingham 001C' 'Birmingham 001D' ...
 'Wolverhampton 035H' 'Wolverhampton 035I' 'Wolverhampton 035J']
```

```
print("Unique LSOA Codes:", unique_lsoa_codes)
```

```
Unique LSOA Codes: ['01009418' '01009419' '01009433' ... '01010530' '01034314' '01034315']
```

```
# Count occurrences of each LSOA code
lsoa_code_counts = df_clean['LSOA code'].value_counts()

# Count occurrences of each LSOA name
```

```
lsoa_name_counts = df_clean['LSOA name'].value_counts()

print(lsoa_code_counts.head())  # show top 5 most common LSOA codes
print(lsoa_name_counts.head())  # show top 5 most common LSOA names
```

```
LSOA code
01033620    374
01033615    197
01010368    169
01034313    157
01010109    146
Name: count, dtype: int64
LSOA name
Birmingham 138A       374
Birmingham 135C       197
Walsall 030A          169
Wolverhampton 020H    157
Solihull 009A         146
Name: count, dtype: int64
```

```
# Example of aggregating data to create a 'Count' column
if 'LSOA name' in df_clean.columns:
    df_count = df_clean['LSOA name'].value_counts().reset_index()
    df_count.columns = ['LSOA name', 'Count']
    print(df_count.head())
```

```
             LSOA name  Count
0     Birmingham 138A    374
1     Birmingham 135C    197
2        Walsall 030A    169
3  Wolverhampton 020H    157
4       Solihull 009A    146
```

```
# Adding a new 'Count' column if it doesn't exist
# Assuming df is your main DataFrame and df_count contains the count information
if 'Count' not in df_clean.columns:
    df_clean = df_clean.merge(df_count, on='LSOA name', how='left')
```

```
from sklearn.feature_extraction.text import CountVectorizer

# Initialize the vectorizer
vectorizer = CountVectorizer()

# Fit and transform the LSOA names
X = vectorizer.fit_transform(df_clean['LSOA name'])

# Convert the result to a DataFrame for better readability
bagofw_df = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out())

# Optionally, add the count to see the frequencies next to the BoW output
bagofw_df['Count'] = df_clean['Count']

print(bagofw_df)
```

```
       001a  001b  001c  001d  001e  001f  001g  001h  002a  002b  ...  on  \
0         0     1     0     0     0     0     0     0     0     0  ...   0
1         0     1     0     0     0     0     0     0     0     0  ...   0
2         0     1     0     0     0     0     0     0     0     0  ...   0
3         0     1     0     0     0     0     0     0     0     0  ...   0
4         0     1     0     0     0     0     0     0     0     0  ...   0
...     ...   ...   ...   ...   ...   ...   ...   ...   ...   ...  ...  ..
25708     0     0     0     0     0     0     0     0     0     0  ...   0
25709     0     0     0     0     0     0     0     0     0     0  ...   0
25710     0     0     0     0     0     0     0     0     0     0  ...   0
25711     0     0     0     0     0     0     0     0     0     0  ...   0
25712     0     0     0     0     0     0     0     0     0     0  ...   0

       rugby  sandwell  solihull  south  staffordshire  stratford  walsall  \
0          0         0         0      0              0          0        0
1          0         0         0      0              0          0        0
2          0         0         0      0              0          0        0
3          0         0         0      0              0          0        0
4          0         0         0      0              0          0        0
...      ...       ...       ...    ...            ...        ...      ...
25708      0         0         0      0              0          0        0
25709      0         0         0      0              0          0        0
25710      0         0         0      0              0          0        0
25711      0         0         0      0              0          0        0
25712      0         0         0      0              0          0        0

       wolverhampton  Count
0                  0      6
1                  0      6
2                  0      6
3                  0      6
4                  0      6
...              ...    ...
25708              1     13
25709              1      4
25710              1      4
25711              1      4
25712              1      4

[25713 rows x 759 columns]
```

```
semi_df = pd.concat([Cmerged_df, bgow_df], axis=1)
```

```
# Drop the unnecessary columns
semi_df = semi_df.drop(columns=['Crime type'])
```

```
semi_df
```

	Location	LSOA code	Zone	Count	action	another	awaiting	case	caution	charged	...	possession	public	robbery	sexual	shoplifting	the	theft	vehicle	violence	weapons
0	Badgers Bank Road	01009418	9	6	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
1	Hook Drive	01009418	9	6	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	1	0
2	Byron Court	01009418	9	6	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	1	0
3	Harcourt Drive	01009418	9	6	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	1	0
4	Hook Drive	01009418	9	6	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
25708	Furnace Drive	01034314	7	13	1	1	0	0	0	0	...	0	0	0	1	0	0	0	0	1	0
25709	Pond Crescent	01034315	7	4	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	0

```
from sklearn.feature_extraction.text import CountVectorizer

# Define your custom list of stop words
custom_stop_words = ['no', 'to', 'by', 'is', 'not', 'in', 'the','be','as']

# Initialize the CountVectorizer with your custom stop words
vectorizer = CountVectorizer(stop_words=custom_stop_words)

# Fit and transform the data
X = vectorizer.fit_transform(df_clean['Last outcome category'])

# Convert to DataFrame for better readability
bagow_df = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out())

print(bagow_df)
```

```
       action  another  awaiting  case  caution  charged  complete  court  \
0           0        0         0     0        0        0         1      0
1           0        0         0     0        0        0         0      0
2           0        0         0     0        0        0         0      0
3           0        0         0     0        0        0         0      0
4           0        0         0     0        0        0         0      0
...       ...      ...       ...   ...      ...      ...       ...    ...
25708       1        1         0     0        0        0         0      0
25709       0        0         0     0        0        0         0      0
25710       0        0         0     0        0        0         1      0
25711       0        0         0     0        0        0         1      0
25712       0        0         0     0        0        0         0      0

       formal  further  ...  part  prosecute  public  resolution  status  \
0           0        0  ...     0          0       0           0       0
1           0        0  ...     0          1       0           0       0
2           0        0  ...     0          1       0           0       0
```

```
3          0        0  ...     0          1       0           0       0
4          0        0  ...     0          1       0           0       0
...      ...      ...  ...   ...        ...     ...         ...     ...
25708      0        0  ...     0          0       0           0       0
25709      0        0  ...     0          1       0           0       0
25710      0        0  ...     0          0       0           0       0
25711      0        0  ...     0          0       0           0       0
25712      0        0  ...     0          0       0           0       1

       suspect  taken  unable  unavailable  update
0            1      0       0            0       0
1            1      0       1            0       0
2            1      0       1            0       0
3            1      0       1            0       0
4            1      0       1            0       0
...        ...    ...     ...          ...     ...
25708        0      1       0            0       0
25709        1      0       1            0       0
25710        1      0       0            0       0
25711        1      0       0            0       0
25712        0      0       0            1       1

[25713 rows x 29 columns]
```

```
unique_location = semi_df['Location'].unique()
unique_location
```

```
array(['Badgers Bank Road', 'Hook Drive', 'Byron Court', ...,
       'Eureka Gardens', 'Pond Crescent', 'Thompson Avenue'], dtype=object)
```

```
# Count occurrences of each LSOA code
Location_counts = semi_df['Location'].value_counts()
Location_counts
```

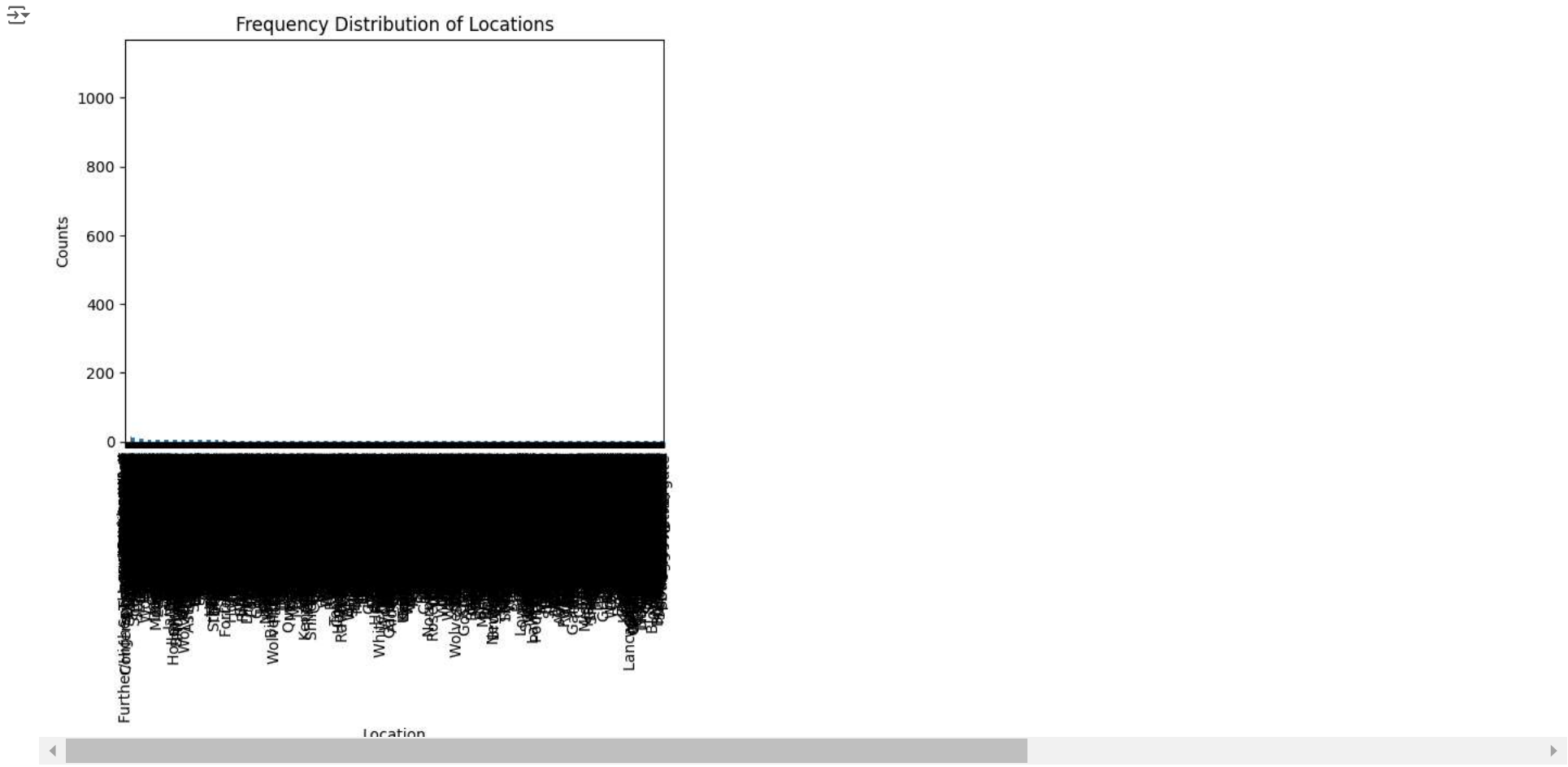
Location	count
Parking Area	1113
Supermarket	1075
Shopping Area	645
Petrol Station	642
Hospital	223
...	...
Pale Street	1
Nethergate	1
Vale Avenue	1
Ox Street	1
Badgers Bank Road	1

8496 rows × 1 columns

```
import pandas as pd

# Assuming you already have a DataFrame 'df' with a 'Location' column
location_counts = df['Location'].value_counts()

print(location_counts)
```

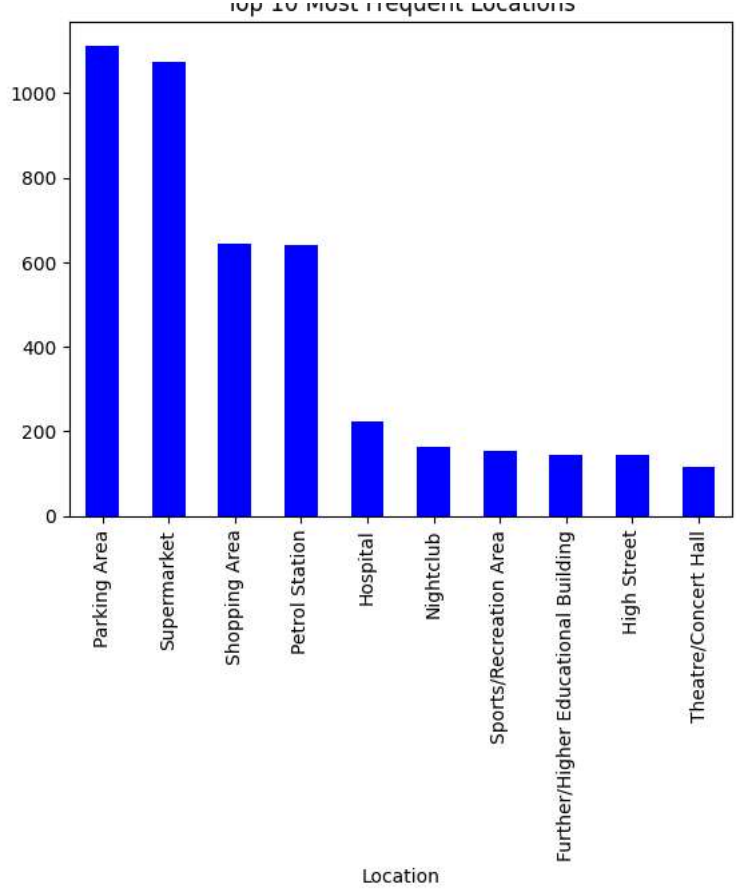
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-35-7d1aae6737a5> in <cell line: 4>()
      2
      3 # Assuming you already have a DataFrame 'df' with a 'Location' column
----> 4 location_counts = df['Location'].value_counts()
      5
      6 print(location_counts)

NameError: name 'df' is not defined
```

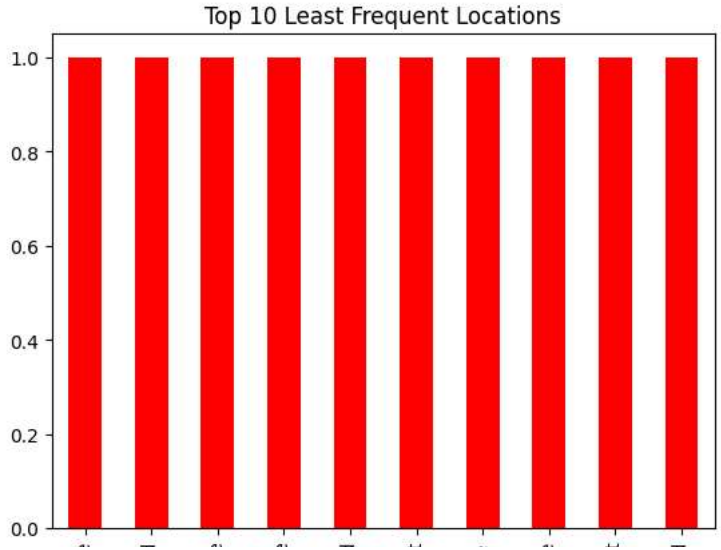
```
import matplotlib.pyplot as plt

Location_counts.plot(kind='bar')
plt.title('Frequency Distribution of Locations')
plt.xlabel('Location')
plt.ylabel('Counts')
plt.show()
```




```
# Plot top 10 most frequent locations
Location_counts.head(10).plot(kind='bar', color='blue')
plt.title('Top 10 Most Frequent Locations')
plt.show()

# Plot 10 least frequent locations
Location_counts.tail(10).plot(kind='bar', color='red')
plt.title('Top 10 Least Frequent Locations')
plt.show()
```

[4]


1000
800
600
400
200
0
Parking Area
Supermarket
Shopping Area
Petrol Station
Hospital
Nightclub
Sports/Recreation Area
Further/Higher Educational Building
High Street
Theatre/Concert Hall
Location


Top 10 Least Frequent Locations
1.0
0.8
0.6
0.4
0.2
0.0


```
# Identify locations considered as outliers based on appearing fewer than 5 times
outlier_locations = Location_counts[Location_counts < 10]
print("Outlier Locations:", outlier_locations)
```

```
Outlier Locations: Location
Summerfield Crescent    9
Old Station Road        9
Asholme Close           9
St Paul'S Street        9
Darwall Street          9
                       ..
Pale Street             1
Nethergate              1
Vale Avenue             1
Ox Street               1
Badgers Bank Road       1
Name: count, Length: 8281, dtype: int64
```

```
outlier_locations.count()
```

```
8281
```

```
# Fit and transform the data
X = vectorizer.fit_transform(semi_df['Location'])

# Convert to DataFrame for better readability
bw_df = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out())

print(bw_df)
```

```
        a34  a4033  a4034  a4037  a4039  a4040  a4041  a4092  a4098  a41  ...  \
0         0      0      0      0      0      0      0      0      0    0  ...
1         0      0      0      0      0      0      0      0      0    0  ...
2         0      0      0      0      0      0      0      0      0    0  ...
3         0      0      0      0      0      0      0      0      0    0  ...
4         0      0      0      0      0      0      0      0      0    0  ...
...     ...    ...    ...    ...    ...    ...    ...    ...    ...  ...  ...
25708     0      0      0      0      0      0      0      0      0    0  ...
25709     0      0      0      0      0      0      0      0      0    0  ...
25710     0      0      0      0      0      0      0      0      0    0  ...
25711     0      0      0      0      0      0      0      0      0    0  ...
25712     0      0      0      0      0      0      0      0      0    0  ...

        yew  yewdale  yewtree  york  yorkswood  young  yoxall  zaria  zion  \
0         0        0        0     0          0      0       0      0     0
1         0        0        0     0          0      0       0      0     0
2         0        0        0     0          0      0       0      0     0
```

```
3       0       0       0    0         0     0      0     0    0
4       0       0       0    0         0     0      0     0    0
...   ...     ...     ...  ...       ...   ...    ...   ...  ...
25708   0       0       0    0         0     0      0     0    0
25709   0       0       0    0         0     0      0     0    0
25710   0       0       0    0         0     0      0     0    0
25711   0       0       0    0         0     0      0     0    0
25712   0       0       0    0         0     0      0     0    0

       zoar
0         0
1         0
2         0
3         0
4         0
...     ...
25708     0
25709     0
25710     0
25711     0
25712     0

[25713 rows x 5774 columns]
```

```
# Concatenate along columns
md_df = pd.concat([semi_df, bw_df], axis=1)
md_df
```

	Location	LSOA code	Zone	Count	action	another	awaiting	case	caution	charged	...	yew	yewdale	yewtree	york	yorkswood	young	yoxall	zaria	zion	zoar
0	Badgers Bank Road	01009418	9	6	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	Hook Drive	01009418	9	6	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	Byron Court	01009418	9	6	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	Harcourt Drive	01009418	9	6	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	Hook Drive	01009418	9	6	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
25708	Furnace Drive	01034314	7	13	1	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
25709	Pond Crescent	01034315	7	4	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
25710	Thompson Avenue	01034315	7	4	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
25711	Thompson Avenue	01034315	7	4	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
25712	Thompson Avenue	01034315	7	4	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

25713 rows × 6589 columns

```
Column_to_drop = ['Location']
md_df = md_df.drop(columns=Column_to_drop)


md_df
```

	LSOA code	Zone	Count	action	another	awaiting	case	caution	charged	complete	...	yew	yewdale	yewtree	york	yorkswood	young	yoxall	zaria	zion	zoar
0	01009418	9	6	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
1	01009418	9	6	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	01009418	9	6	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	01009418	9	6	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	01009418	9	6	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
25708	01034314	7	13	1	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
25709	01034315	7	4	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
25710	01034315	7	4	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
25711	01034315	7	4	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
25712	01034315	7	4	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

25713 rows × 6588 columns

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier


# 'X' contains all columns except 'LSOA code' and 'Zone', which is our target
X = md_df.drop(['Zone'], axis=1)
y = md_df['Zone']


# Split data for classification
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

from sklearn.metrics import classification_report, confusion_matrix
from sklearn.ensemble import RandomForestClassifier

# Initialize and train the Random Forest classifier
classifier = RandomForestClassifier(n_estimators=100, random_state=42)
classifier.fit(X_train, y_train)

# Evaluate the classifier
classification_score = classifier.score(X_test, y_test)
print(f'Classification Model Accuracy: {classification_score:.2f}')


# Make predictions
y_pred = classifier.predict(X_test)

# Generate classification report and confusion matrix
print(classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Classification Model Accuracy: 0.96
              precision    recall  f1-score   support
```

```
              0       0.96      0.98      0.97       725
              1       0.96      0.99      0.97       649
              2       1.00      0.99      1.00       896
              3       0.98      0.95      0.96      1283
              4       0.92      0.96      0.94       674
              5       0.95      0.94      0.95       796
              6       0.94      0.94      0.94       681
              7       0.99      0.97      0.98       833
              8       0.94      0.99      0.96       526
              9       0.97      0.95      0.96       651

    accuracy                           0.96      7714
   macro avg       0.96      0.97      0.96      7714
weighted avg       0.96      0.96      0.96      7714

Confusion Matrix:
 [[ 711    0    0    0   11    0    0    3    0    0]
 [   0  640    0    1    0    2    5    0    0    1]
 [   0    3  891    0    0    0    0    2    0    0]
 [   7    0    0 1217   31   13   11    0    0    4]
 [   9    0    0    0  648    0    1    2   14    0]
 [   0   22    0    8    0  748   13    0    0    5]
 [   0    0    0    8   10   17  639    0    0    7]
 [  10    0    0    0    1    0    0  804   18    0]
 [   0    0    0    0    7    0    0    0  519    0]
 [   6    0    0   10    0    6    8    0    0  621]]
```

X_test

	LSOA code	Count	action	another	awaiting	case	caution	charged	complete	court	...	yew	yewdale	yewtree	york	yorkswood	young	yoxall	zaria	zion	zoar
341	508	53	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
12810	680	56	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5742	4	15	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
22715	1280	24	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
11952	584	18	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7649	371	16	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
12024	670	22	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
6067	328	4	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
10495	1716	17	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
20453	1214	72	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0

7714 rows × 6587 columns

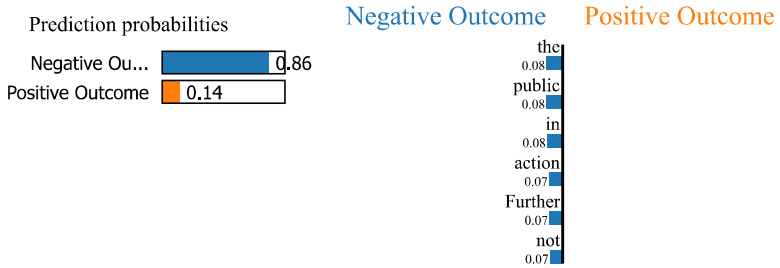
```
import lime
from lime.lime_text import LimeTextExplainer
from sklearn.pipeline import Pipeline
```

कोडिंग शुरू करें या एआई की मदद से कोड जनरेट करें.

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-77-59abc4ca3d01> in <cell line: 2>()
      1 ## define the pipeline
      2 pipeline = Pipeline([
----> 3   ('vectorizer',tfidf_vec),
      4   ('clf', svm.SVC(C=1, kernel='linear', probability=True))])
      5

NameError: name 'tfidf_vec' is not defined
```

```
import textwrap
reviews_test = X_test
sentiments_test = y_test



# We choose a sample from test set
idx = 210
text_sample = reviews_test[idx]
class_names = ['negative', 'positive']

print('Review ID-{}:'.format(idx))
print('-'*50)
print('Review Text:\n', textwrap.fill(text_sample,400))
print('-'*50)
print('Probability(positive) =', pipeline.predict_proba([text_sample])[0,1])
print('Probability(negative) =', pipeline.predict_proba([text_sample])[0,0])
print('Predicted class: %s' % pipeline.predict([text_sample]))
print('True class: %s' % sentiments_test[idx])


import matplotlib
matplotlib.rcParams['figure.dpi']=300
%matplotlib inline


explainer = LimeTextExplainer(class_names=class_names)
explanation = explainer.explain_instance(text_sample,
                                         pipeline.predict_proba,
                                         num_features=20)
explanation.show_in_notebook(text=True)


import lime
from lime.lime_text import LimeTextExplainer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer
import textwrap
import matplotlib
matplotlib.rcParams['figure.dpi'] = 300
import pandas as pd

# Sample DataFrame (replace this with your actual DataFrame)
data = {'Last outcome category': [
    'Unable to prosecute suspect', 'Investigation complete; no suspect identified',
```

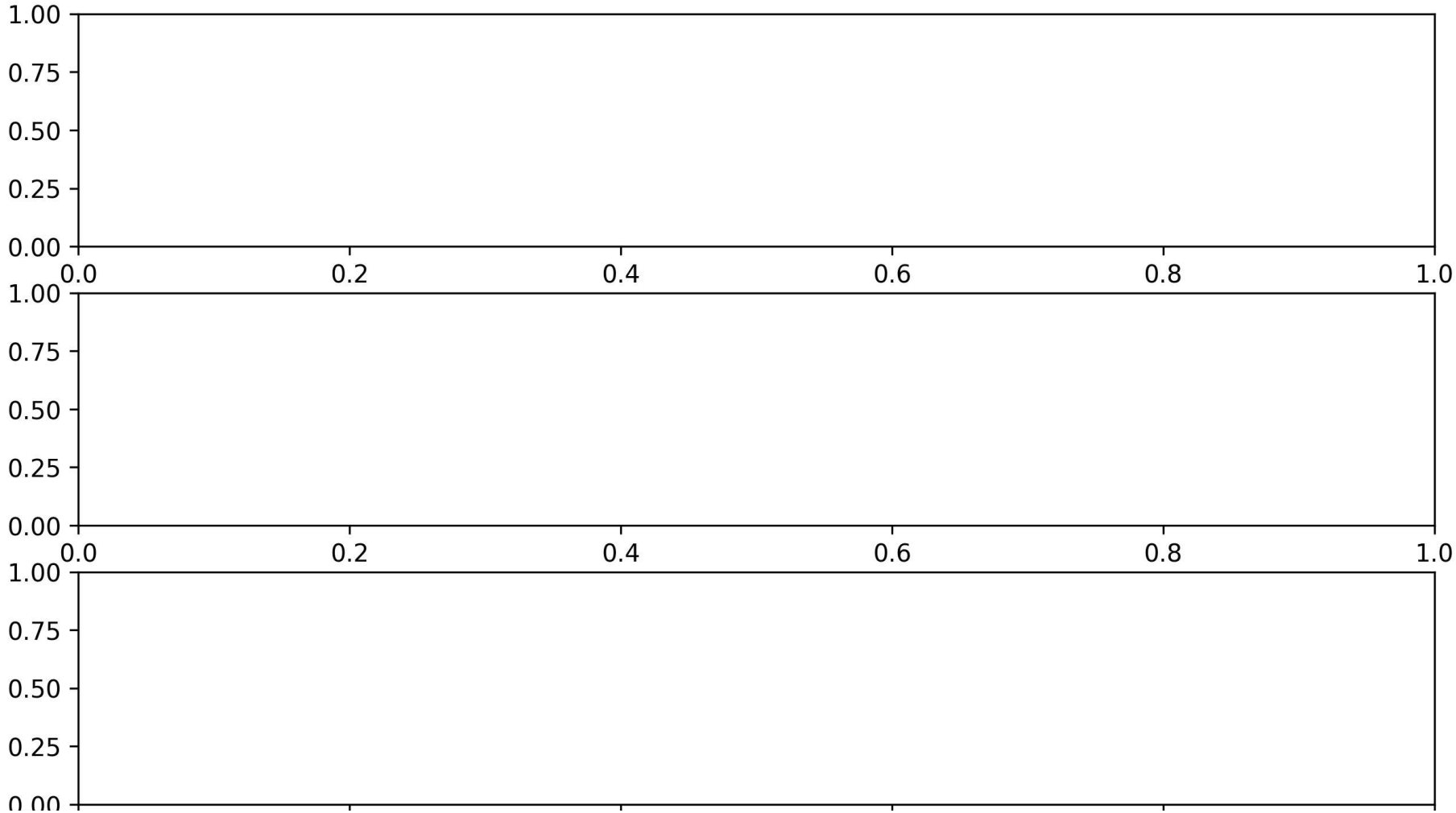
```
        'Further action is not in the public interest', 'Formal action is not in the public interest',
        'Offender given a caution', 'Local resolution'
    ], 'Outcome Label': [0, 1, 0, 0, 1, 1]}  # 0=Negative outcome, 1=Positive outcome

    df = pd.DataFrame(data)

    # Features and Target
    X_train = df['Last outcome category']
    y_train = df['Outcome Label']

    # Vectorization (CountVectorizer or TfidfVectorizer)
    vectorizer = CountVectorizer()

    # RandomForestClassifier pipeline
    pipeline = Pipeline([
        ('vectorizer', vectorizer),
        ('clf', RandomForestClassifier())
    ])

    # Train the model
    pipeline.fit(X_train, y_train)

    # Instantiate LIME explainer
    explainer = LimeTextExplainer(class_names=['Negative Outcome', 'Positive Outcome'])

    # Sample Text from test data
    idx = 2  # Test on a specific index, you can adjust this
    text_sample = X_train[idx]

    # Explain the prediction for this instance
    explanation = explainer.explain_instance(text_sample,
                                             pipeline.predict_proba,
                                             num_features=6)

    # Show explanation
    explanation.show_in_notebook(text=True)

    # Print the text sample and prediction probabilities
    print(f'Review ID-{idx}:')
    print('-'*50)
    print('Text Sample:\n', textwrap.fill(text_sample, 400))
    print('-'*50)
    print('Probability(Positive) =', pipeline.predict_proba([text_sample])[0, 1])
    print('Probability(Negative) =', pipeline.predict_proba([text_sample])[0, 0])
    print('Predicted class: %s' % pipeline.predict([text_sample]))
    print('True class: %s' % y_train[idx])
```

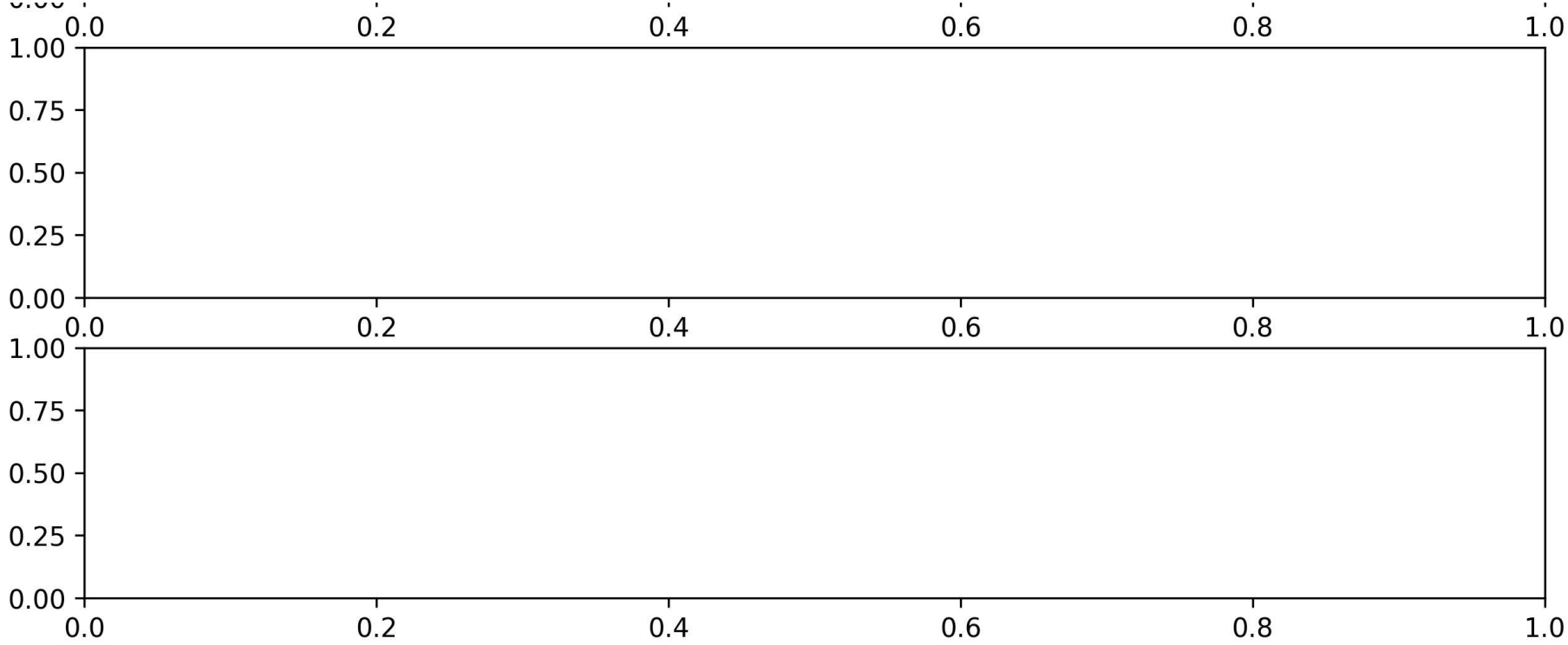



```
Review ID-2:
--------------------------------------------------
Text Sample:
 Further action is not in the public interest
--------------------------------------------------
Probability(Positive) = 0.14
Probability(Negative) = 0.86
Predicted class: [0]
True class: 0
```

```
import lime
from lime.lime_text import LimeTextExplainer
import matplotlib.pyplot as plt

# Assuming 'pipeline' has been defined and fitted as shown in previous examples
# Assuming 'df_clean' is your DataFrame and 'Last outcome category' is the column with texts

explainer = LimeTextExplainer(class_names=['Negative Outcome', 'Positive Outcome'])

# Select how many instances you want to explain, here I use just 5 for demonstration
n_samples = 5
fig, axs = plt.subplots(n_samples, figsize=(10, n_samples * 2))

for idx in range(n_samples):
    text_instance = df_clean['Last outcome category'].iloc[idx]
    explanation = explainer.explain_instance(text_instance, pipeline.predict_proba, num_features=6)

    # Visualizing the explanation
    explanation.as_pyplot_figure(ax=axs[idx])
    axs[idx].set_title(f'Review ID-{idx}: {text_instance[:30]}...')  # Title with part of the text

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

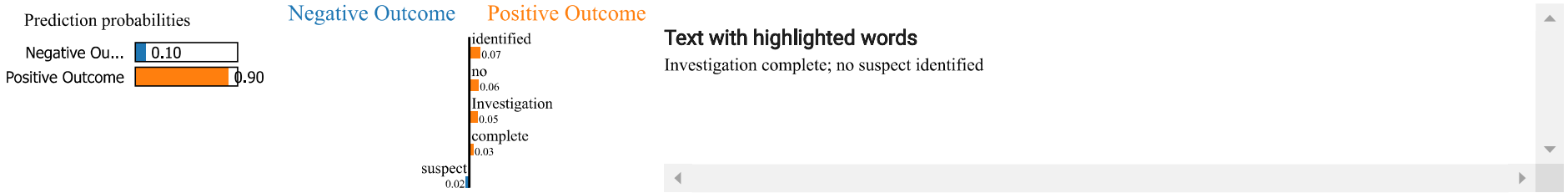
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-81-64b03a125788> in <cell line: 14>()
     17
     18     # Visualizing the explanation
---> 19     explanation.as_pyplot_figure(ax=axs[idx])
     20     axs[idx].set_title(f'Review ID-{idx}: {text_instance[:30]}...')  # Title with part of the text
     21

                                  1 frames
/usr/local/lib/python3.10/dist-packages/lime/explanation.py in as_list(self, label, **kwargs)
    139         """
    140         label_to_use = label if self.mode == "classification" else self.dummy_label
--> 141         ans = self.domain_mapper.map_exp_ids(self.local_exp[label_to_use], **kwargs)
    142         ans = [(x[0], float(x[1])) for x in ans]
    143         return ans

TypeError: TextDomainMapper.map_exp_ids() got an unexpected keyword argument 'ax'
```




0.0
0.2
0.4
0.6
0.8
1.0
1.00
0.75
0.50
0.25
0.00

```
import lime
from lime.lime_text import LimeTextExplainer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer
import textwrap
import matplotlib
matplotlib.rcParams['figure.dpi'] = 300
import pandas as pd

# Sample DataFrame (replace this with your actual DataFrame)
data = {'Last outcome category': [
    'Unable to prosecute suspect', 'Investigation complete; no suspect identified',
    'Further action is not in the public interest', 'Formal action is not in the public interest',
    'Offender given a caution', 'Local resolution'
], 'Outcome Label': [0, 1, 0, 0, 1, 1]}  # 0=Negative outcome, 1=Positive outcome

df = pd.DataFrame(data)

# Features and Target
X_train = df['Last outcome category']
y_train = df['Outcome Label']

# Vectorization (CountVectorizer or TfidfVectorizer)
vectorizer = CountVectorizer()

# RandomForestClassifier pipeline
pipeline = Pipeline([
    ('vectorizer', vectorizer),
    ('clf', RandomForestClassifier())
])

# Train the model
pipeline.fit(X_train, y_train)

# Instantiate LIME explainer
explainer = LimeTextExplainer(class_names=['Negative Outcome', 'Positive Outcome'])

# Sample Text from test data
idx = 1  # Test on a specific index, you can adjust this
text_sample = X_train[idx]

# Explain the prediction for this instance
explanation = explainer.explain_instance(text_sample,
                                         pipeline.predict_proba,
                                         num_features=6)

# Show explanation
explanation.show_in_notebook(text=True)

# Print the text sample and prediction probabilities
print(f'Review ID-{idx}:')
print('-'*50)
print('Text Sample:\n', textwrap.fill(text_sample, 400))
print('-'*50)
print('Probability(Positive) =', pipeline.predict_proba([text_sample])[0, 1])
print('Probability(Negative) =', pipeline.predict_proba([text_sample])[0, 0])
```

```
print('Predicted class: %s' % pipeline.predict([text_sample]))
print('True class: %s' % y_train[idx])
```




```
Review ID-1:
--------------------------------------------------
Text Sample:
 Investigation complete; no suspect identified
--------------------------------------------------
Probability(Positive) = 0.9
Probability(Negative) = 0.1
Predicted class: [1]
True class: 1
```

```
import lime
import lime.lime_tabular
import numpy as np
import pandas as pd

# Assuming you have already trained your RandomForestClassifier
# 'classifier' is your trained RandomForestClassifier
# 'X_test' is your test data in a dense format (if it's sparse, convert to dense with .toarray())
# 'y_test' are your target labels

# Ensure that the test data is in a dense format if it was in a sparse matrix (e.g., from CountVectorizer)
X_test_dense = X_test.toarray() if hasattr(X_test, 'toarray') else X_test

# Define feature names for LIME
feature_names = X.columns.tolist()  # or use the correct feature names for your dataset

# Initialize the LIME explainer
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_train.values,  # Your training data in dense format
    feature_names=feature_names,  # Feature names of your dataset
    class_names=['class1', 'class2'],  # Adjust based on your classification problem
    mode='classification'  # Use classification for RandomForestClassifier
)

# Select an instance from the test data to explain
idx = 0  # Change this index to select other instances
instance = X_test_dense[idx].reshape(1, -1)

# Explain the prediction for this instance
exp = explainer.explain_instance(
    data_row=instance[0],  # The specific data row you're explaining
    predict_fn=classifier.predict_proba,  # The RandomForest prediction probability function
    num_features=10  # Focus on the top 10 features (or adjust as needed)
)
```

```
# Display the explanation in a readable format
exp.show_in_notebook(show_table=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3790         try:
-> 3791             return self._engine.get_loc(casted_key)
   3792         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine._get_loc_duplicates()

index.pyx in pandas._libs.index.IndexEngine._maybe_get_bool_indexer()

index.pyx in pandas._libs.index._unpack_bool_indexer()

KeyError: 0

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
                                          2 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3796             ):
   3797                 raise InvalidIndexError(key)
-> 3798             raise KeyError(key) from err
   3799         except TypeError:
   3800             # If we have a listlike key, _check_indexing_error will raise

KeyError: 0
```

```
X_test

explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=md_df.toarray(),  # Ensure the data is in a dense format if it was sparse
    feature_names=feature_names,  # Names of the features in your dataset
    class_names=['class1', 'class2'],  # Adjust based on your classification problem
    mode='classification'  # or 'regression' if it's a regression problem
)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-74-fa5d8222a0dd> in <cell line: 3>()
      2
      3 explainer = lime.lime_tabular.LimeTabularExplainer(
----> 4     training_data=md_df.toarray(),  # Ensure the data is in a dense format if it was sparse
      5     feature_names=feature_names,  # Names of the features in your dataset
      6     class_names=['class1', 'class2'],  # Adjust based on your classification problem

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in __getattr__(self, name)
   6202         ):
   6203             return self[name]
-> 6204         return object.__getattribute__(self, name)
   6205
   6206     @final

AttributeError: 'DataFrame' object has no attribute 'toarray'
```

```
# Assuming 'classifier' is your trained RandomForestClassifier
importances = classifier.feature_importances_
feature_names = X.columns

# Create a DataFrame to view the features and their importance scores
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
print(feature_importance_df.head(10))  # Display top 10 important features
```

```
           Feature  Importance
0        LSOA code    0.117562
776       coventry    0.067978
788  wolverhampton    0.052825
774     birmingham    0.048393
787        walsall    0.046421
782       sandwell    0.041180
789          Count    0.036467
777         dudley    0.036433
1            Count    0.034661
783       solihull    0.025218
```

```
import lime
import lime.lime_tabular
import numpy as np

# Assuming 'X_test' is your test dataset and 'y_test' is the corresponding labels
# 'classifier' is your trained RandomForestClassifier
# 'feature_names' is a list of all feature names used in the RandomForest model

# Initialize the LIME explainer for tabular data
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_test.toarray(),  # Ensure the data is in a dense format if it was sparse
    feature_names=feature_names,  # Names of the features in your dataset
    class_names=['class1', 'class2'],  # Adjust based on your classification problem
    mode='classification'  # or 'regression' if it's a regression problem
)

# Select an instance to explain, here 'idx' is the index of the instance in 'X_test'
```

```
idx = 0
instance = X_test[idx].reshape(1, -1)

# Generate an explanation for the prediction of this instance
exp = explainer.explain_instance(
    data_row=instance[0],  # instance must be passed as a 1D array
    predict_fn=classifier.predict_proba,  # Prediction function of the RandomForest model
    num_features=10  # focusing on the top 10 features, or adjust as per your specific case
)

# Display the explanation
exp.show_in_notebook(show_table=True)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-69-27283a060ea0> in <cell line: 10>()
      9 # Initialize the LIME explainer for tabular data
     10 explainer = lime.lime_tabular.LimeTabularExplainer(
---> 11     training_data=X_test.toarray(),  # Ensure the data is in a dense format if it was sparse
     12     feature_names=feature_names,  # Names of the features in your dataset
     13     class_names=['class1', 'class2'],  # Adjust based on your classification problem

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in __getattr__(self, name)
   6202         ):
   6203             return self[name]
-> 6204         return object.__getattribute__(self, name)
   6205
   6206     @final

AttributeError: 'DataFrame' object has no attribute 'toarray'
```

कोडिंग शुरू करें या एआई की मदद से कोड जनरेट करें.

```
import lime
import lime.lime_tabular
import numpy as np

# Assume 'classifier' is your trained RandomForestClassifier
# and 'vectorizer' is your fitted CountVectorizer

# Preparing the explainer
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data= bgow_df.toarray(),  # convert sparse matrix to dense
    feature_names=vectorizer.get_feature_names_out(),  # or however you have named your features
    class_names=['Negative', 'Positive'],  # Adjust based on your actual classes
    mode='classification'
)

# Choose an instance to explain
idx = 1  # Index of the instance in your dataframe
instance = md_df[idx].toarray()  # Make sure it's in the correct format

# Generate the explanation
explanation = explainer.explain_instance(
    instance[0],  # The instance must be passed as a 1D array
```

```
    classifier.predict_proba,  # The prediction method of your RandomForest
    num_features=10  # Number of top features to include in the explanation
)

# Display the explanation
explanation.show_in_notebook(show_table=True)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-68-975d5f7bfd34> in <cell line: 9>()
      8 # Preparing the explainer
      9 explainer = lime.lime_tabular.LimeTabularExplainer(
---> 10     training_data= bgow_df.toarray(),  # convert sparse matrix to dense
     11     feature_names=vectorizer.get_feature_names_out(),  # or however you have named your features
     12     class_names=['Negative', 'Positive'],  # Adjust based on your actual classes

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in __getattr__(self, name)
   6202         ):
   6203             return self[name]
-> 6204         return object.__getattribute__(self, name)
   6205
   6206     @final

AttributeError: 'DataFrame' object has no attribute 'toarray'
```

```
# Vectorize the instance using the already fitted vectorizer
instance_vectorized = vectorizer.transform([df_clean['Last outcome category'][idx]])

# Explain using the vectorized instance
exp = explainer.explain_instance(instance_vectorized.toarray()[0],
                                 classifier.predict_proba,
                                 num_features=6)
exp.show_in_notebook(text=False)  # Set text=False since data is already in vector form
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-53-60434e107202> in <cell line: 5>()
      3
      4 # Explain using the vectorized instance
----> 5 exp = explainer.explain_instance(instance_vectorized.toarray()[0],
      6                                  classifier.predict_proba,
      7                                  num_features=6)

                                  1 frames
/usr/local/lib/python3.10/dist-packages/lime/lime_text.py in __init__(self, raw_string, split_expression, bow, mask_string)
    112             # the separator character from the split results.
    113             splitter = re.compile(r'(%s)|$' % split_expression)
--> 114             self.as_list = [s for s in splitter.split(self.raw) if s]
    115             non_word = splitter.match
    116

TypeError: cannot use a string pattern on a bytes-like object
```

```
md_df
```

	LSOA code	Zone	Count	action	another	awaiting	case	caution	charged	complete	...	yew	yewdale	yewtree	york	yorkswood	young	yoxall	zaria	zion	zoar
0	478	9	6	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
1	478	9	6	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	478	9	6	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	478	9	6	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	478	9	6	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
25708	1660	7	13	1	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
25709	1661	7	4	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
25710	1661	7	4	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
25711	1661	7	4	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
25712	1661	7	4	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

25713 rows × 6588 columns

```
from lime.lime_text import LimeTextExplainer
from sklearn.pipeline import make_pipeline
import sklearn.ensemble
import sklearn.feature_extraction

# Vectorizing the dataset
vectorizer = sklearn.feature_extraction.text.CountVectorizer()
X_vectorized = vectorizer.fit_transform(df_clean['Last outcome category'].astype(str))

# Create a pipeline
classifier = sklearn.ensemble.RandomForestClassifier()
pipeline = make_pipeline(vectorizer, classifier)
pipeline.fit(X_vectorized, 'Zone')  # Assuming y is your target array

# Instantiate LIME Explainer
explainer = LimeTextExplainer(class_names=['class1', 'class2'])  # Modify as per your classes

# Explain an instance
idx = 1  # Index of the instance you want to explain
exp = explainer.explain_instance(df_clean['Last outcome category'][idx], pipeline.predict_proba, num_features=6)
exp.show_in_notebook(text=True)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-62-37fa62238218> in <cell line: 13>()
     11 classifier = sklearn.ensemble.RandomForestClassifier()
     12 pipeline = make_pipeline(vectorizer, classifier)
---> 13 pipeline.fit(X_vectorized, 'Zone')  # Assuming y is your target array
     14
     15 # Instantiate LIME Explainer

                                 9 frames
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py in _preprocess(doc, accent_function, lower)
     64     """
     65     if lower:
---> 66         doc = doc.lower()
     67     if accent_function is not None:
     68         doc = accent_function(doc)

AttributeError: 'csr_matrix' object has no attribute 'lower'
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer
from lime.lime_text import LimeTextExplainer
import numpy as np

# Sample data
data = {
    'Last outcome category': [
        'Unable to prosecute suspect', 'Investigation complete; no suspect identified',
        'Status update unavailable', 'Awaiting court outcome',
        'Further action is not in the public interest'
    ],
    'Target': [0, 1, 0, 1, 0]  # Example binary targets
}
df = pd.DataFrame(data)

# Preprocess and vectorize data
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(df_clean['Last outcome category'])

# Train classifier
classifier = RandomForestClassifier(random_state=42)
classifier.fit(X_vectorized, md_df['Zone'])  # Assuming 'Target' is your label column

# Set up Lime for text data
class_names = ['Negative', 'Positive']  # Adjust based on your actual classes
explainer = LimeTextExplainer(class_names=class_names)

# Select an instance to explain
idx = 1
exp = explainer.explain_instance(df_clean['Last outcome category'][idx],
                                 classifier.predict_proba,
                                 num_features=6,
                                 labels=[1])

# Display the explanation
```

```
exp.show_in_notebook(text=True)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-57-2f4cf2558acc> in <cell line: 31>()
     29 # Select an instance to explain
     30 idx = 1
---> 31 exp = explainer.explain_instance(df_clean['Last outcome category'][idx],
     32                                  classifier.predict_proba,
     33                                  num_features=6,

6 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/_array_api.py in _asarray_with_order(array, dtype, order, copy, xp, device)
    743             array = numpy.array(array, order=order, dtype=dtype)
    744         else:
--> 745             array = numpy.asarray(array, order=order, dtype=dtype)
    746
    747         # At this point array is a NumPy ndarray. We convert it to an array

ValueError: could not convert string to float: 'Unable to prosecute suspect'
```

```
import lime
import lime.lime_tabular
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

# Assuming 'classifier' is your trained RandomForestClassifier
importances = classifier.feature_importances_
feature_names = X.columns

# Create a DataFrame to view the features and their importance scores
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
print(feature_importance_df.head(20))  # Display top 10 important features

# Setup Lime explainer
explainer = lime.lime_tabular.LimeTabularExplainer(
    X_train.values,
    feature_names=X_train.columns.tolist(),
    class_names=['Non-Target', 'Target'], # Adjust based on your actual target classes
    mode='classification'
)

# Choose an instance to explain
idx = 20  # Change this index based on your specific instance of interest in X_test
data_instance = X_test.iloc[idx]

# Explain the prediction
exp = explainer.explain_instance(
    data_instance.values,
    classifier.predict_proba,
    num_features=10  # You can adjust the number of features you want to show in the explanation
)
```

```
# Show the explanation in notebook
exp.show_in_notebook(show_table=True)
```

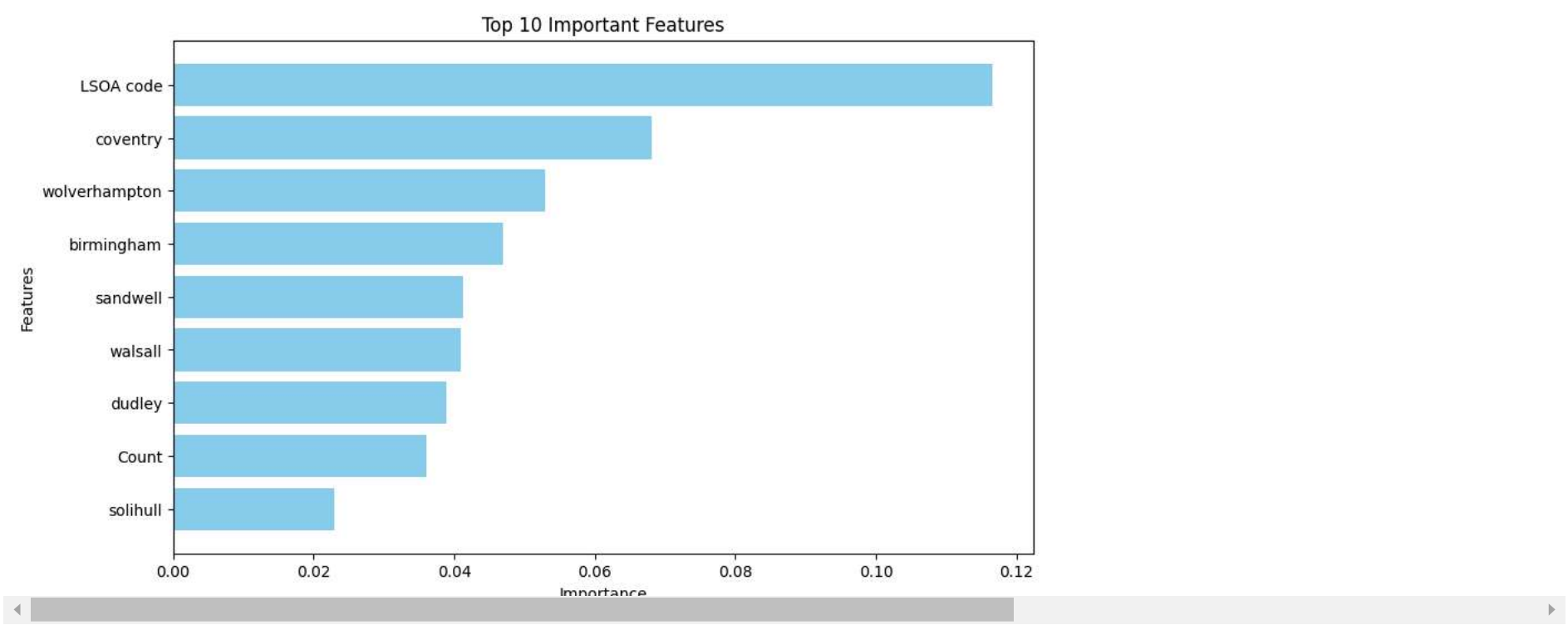
```
            Feature  Importance
0         LSOA code    0.117562
776        coventry    0.067978
788   wolverhampton    0.052825
774      birmingham    0.048393
787         walsall    0.046421
782        sandwell    0.041180
789           Count    0.036467
777          dudley    0.036433
1             Count    0.034661
783        solihull    0.025218
5823         street    0.005417
5288           road    0.004038
741            138a    0.003134
2041          close    0.002563
81             009a    0.002411
486            081f    0.002302
217            030a    0.002231
807     shoplifting    0.002153
799        offences    0.002131
508            087f    0.002119
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-47-72e248ff3638> in <cell line: 18>()
     16
     17 # Setup Lime explainer
---> 18 explainer = lime.lime_tabular.LimeTabularExplainer(
     19     X_train.values,
     20     feature_names=X_train.columns.tolist(),

 9 frames
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py in _lerp(a, b, t, out)
   4653         Output array.
   4654     """
-> 4655     diff_b_a = subtract(b, a)
   4656     # asanyarray is a stop-gap until gh-13105
   4657     lerp_interpolation = asanyarray(add(a, diff_b_a * t, out=out))

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Assuming 'df' is your DataFrame containing the feature and target data
# Convert categorical variables to numeric
label_encoders = {}
categorical_columns = ['LSOA code', 'coventry', 'wolverhampton', 'birmingham', 'walsall', 'sandwell', 'dudley', 'solihull']  # update this list based on your actual categorical features
for col in categorical_columns:
    le = LabelEncoder()
   df_clean[col] = le.fit_transform(df_clean[col])
    label_encoders[col] = le

# Now set up your train-test split, model training, and LIME explanation as before

# Your RandomForestClassifier setup and training here
```

```
# Setup Lime explainer after ensuring all features are numeric
explainer = lime.lime_tabular.LimeTabularExplainer(
    X_train.values,
    feature_names=X_train.columns.tolist(),
    class_names=['Non-Target', 'Target'], # Update based on your actual target classes
    mode='classification'
)

# Select instance to explain
idx = 20  # Adjust index as necessary
data_instance = X_test.iloc[idx]

# Generate the explanation
exp = explainer.explain_instance(
    data_instance.values,
    classifier.predict_proba,
    num_features=10  # Adjust number of features as necessary
)
exp.show_in_notebook(show_table=True)
```

```
  File "<tokenize>", line 10
    df_clean[col] = le.fit_transform(df_clean[col])
    ^
IndentationError: unindent does not match any outer indentation level
```

```
import matplotlib.pyplot as plt

# Plotting feature importances
plt.figure(figsize=(10, 6))
plt.barh(feature_importance_df['Feature'][:10], feature_importance_df['Importance'][:10], color='skyblue')
plt.xlabel('Importance')
plt.ylabel('Features')
plt.title('Top 10 Important Features')
plt.gca().invert_yaxis()  # Invert the Y-axis to show the highest importance at the top
plt.show()
```




```
pip install lime
```

```
Requirement already satisfied: lime in /usr/local/lib/python3.10/dist-packages (0.2.0.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from lime) (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from lime) (1.26.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from lime) (1.13.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from lime) (4.66.5)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.10/dist-packages (from lime) (1.5.2)
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.10/dist-packages (from lime) (0.24.0)
Requirement already satisfied: networkx>=2.8 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.12->lime) (3.3)
Requirement already satisfied: pillow>=9.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.12->lime) (10.4.0)
Requirement already satisfied: imageio>=2.33 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.12->lime) (2.35.1)
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.12->lime) (2024.9.20)
Requirement already satisfied: packaging>=21 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.12->lime) (24.1)
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.12->lime) (0.4)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.18->lime) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.18->lime) (3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->lime) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->lime) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->lime) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->lime) (1.4.7)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->lime) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->lime) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->lime) (1.16.0)
```

```
import numpy as np
from lime.lime_tabular import LimeTabularExplainer
```

```
from sklearn.ensemble import RandomForestClassifier  # Assuming the use of RandomForest


X = md_df.drop('Zone', axis=1).values  # Features
y = md_df['Zone'].values  # Target
feature_names = md_df.drop('Zone', axis=1).columns.tolist()
class_names = [str(c) for c in np.unique(y)]  # Convert class names to string if not already


explainer = LimeTabularExplainer(X,
                                 feature_names=feature_names,
                                 class_names=class_names,
                                 mode='classification')
```

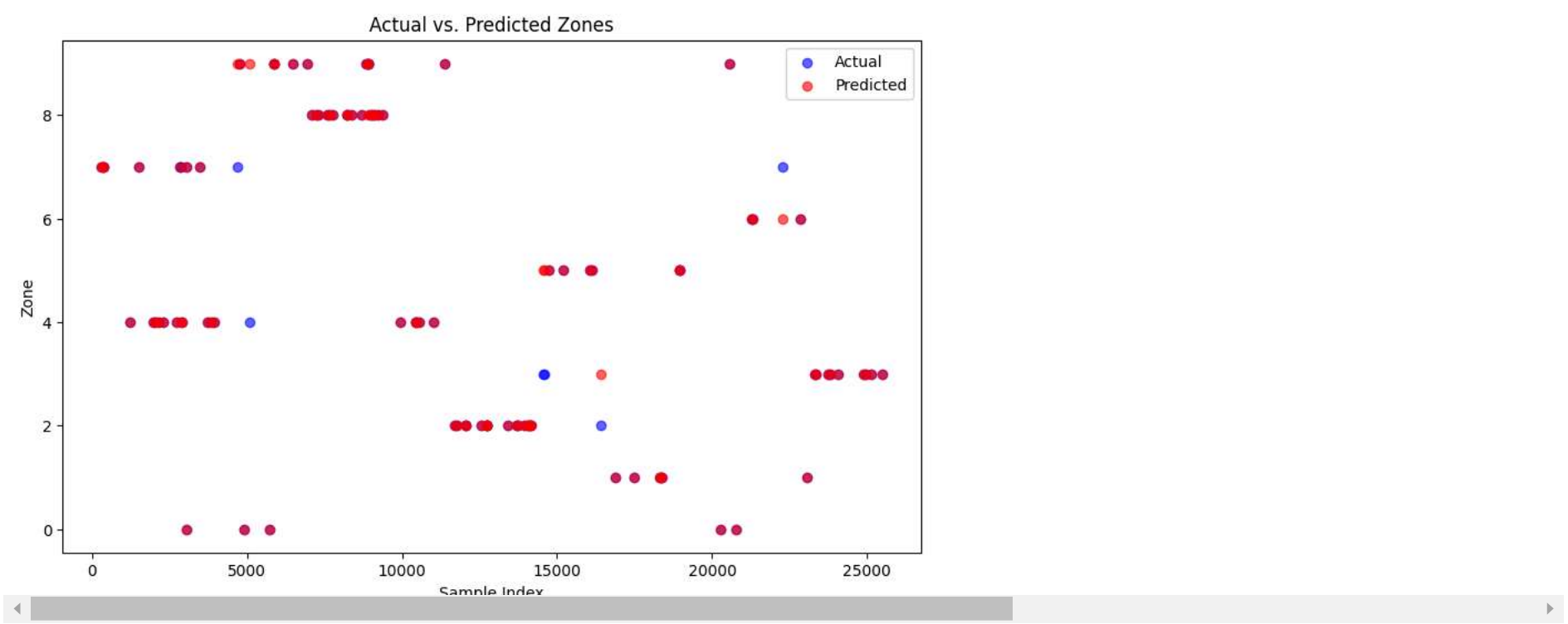
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-42-a063ac627133> in <cell line: 1>()
----> 1 explainer = LimeTabularExplainer(X,
      2                                 feature_names=feature_names,
      3                                 class_names=class_names,
      4                                 mode='classification')

                                 9 frames
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py in _lerp(a, b, t, out)
   4653         Output array.
   4654     """
-> 4655     diff_b_a = subtract(b, a)
   4656     # asanyarray is a stop-gap until gh-13105
   4657     lerp_interpolation = asanyarray(add(a, diff_b_a * t, out=out))

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

```
instance_index = 0
data_instance = X[instance_index]


explanation = explainer.explain_instance(data_instance, rf_model.predict_proba, num_features=10)


explanation.show_in_notebook(show_table=True)
```

कोडिंग शुरू करें या एआई की मदद से कोड जनरेट करें.

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-62-a063ac627133> in <cell line: 1>()
      1 explainer = LimeTabularExplainer(X,
      2                                  feature_names=feature_names,
----> 3                                  class_names=class_names,
      4                                  mode='classification')

NameError: name 'class_names' is not defined
```

```
# Predict on the test set
y_pred = classifier.predict(X_test)

# For visualization, you might want to use a sample or the full data depending on the size
import numpy as np

# Create a DataFrame for easier plotting
results_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

# Sample for plotting if the dataset is too large
sample_df = results_df.sample(n=100, random_state=42)  # Adjust n as necessary

# Plotting actual vs predicted
plt.figure(figsize=(10, 6))
plt.scatter(sample_df.index, sample_df['Actual'], color='blue', label='Actual', alpha=0.6)
plt.scatter(sample_df.index, sample_df['Predicted'], color='red', label='Predicted', alpha=0.6)
plt.title('Actual vs. Predicted Zones')
plt.xlabel('Sample Index')
plt.ylabel('Zone')
plt.legend()
plt.show()
```




```
from sklearn.metrics import log_loss, accuracy_score

# Training predictions for "loss" estimation
y_train_pred = classifier.predict(X_train)
y_test_pred = classifier.predict(X_test)

# Calculating "losses" using accuracy as a simple proxy
train_accuracy = accuracy_score(y_train, y_train_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)

print(f"Training Accuracy: {train_accuracy:.2f}")
print(f"Test Accuracy: {test_accuracy:.2f}")
```

```
Training Accuracy: 1.00
Test Accuracy: 0.96
```

```
from sklearn.metrics import confusion_matrix

conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

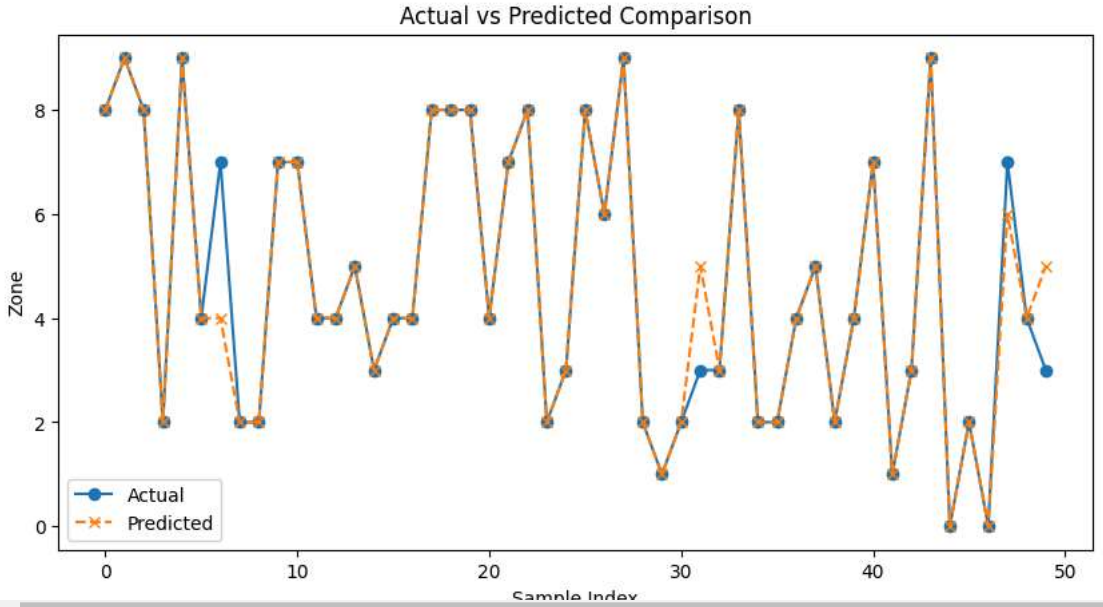
```
Confusion Matrix:
[[ 641    0    0    0    0    0    0    0    3    5]
 [   0  663    0    4    0   14   14    0    0    0]
```

```
[   3    0  891    2    0    0    0    0    0    0]
[   0    1    0  818    0   25    8    0    0    0]
[   0   33    0    0 1224    0    6    3   11    8]
[   0    7    0    0    0  523    0    0    0    0]
[   0   11    0    1    0    0  657    0    0    0]
[   0    0    0    0    8    0    6  627    8    7]
[   0    6    0    0    6    0    0    6  639   20]
[  26    0    0    0    7    0    0    5   15  752]]
```

```
# Creating a DataFrame for actual vs. predicted values
results_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

# Displaying the first few rows of the results
print(results_df.head(10))
```

```
        Actual  Predicted
341          7          7
12810        2          2
5742         0          0
22715        6          6
11952        2          2
16586        6          6
1448         7          7
9838         4          4
15014        5          5
18790        1          1
```

```
# Plotting a sample of the results
plt.figure(figsize=(10, 5))
sample_results = results_df.sample(50, random_state=42)  # Taking a sample for clear visualization
plt.plot(sample_results['Actual'].values, label='Actual', marker='o')
plt.plot(sample_results['Predicted'].values, label='Predicted', linestyle='--', marker='x')
plt.title('Actual vs Predicted Comparison')
plt.ylabel('Zone')
plt.xlabel('Sample Index')
plt.legend()
plt.show()
```

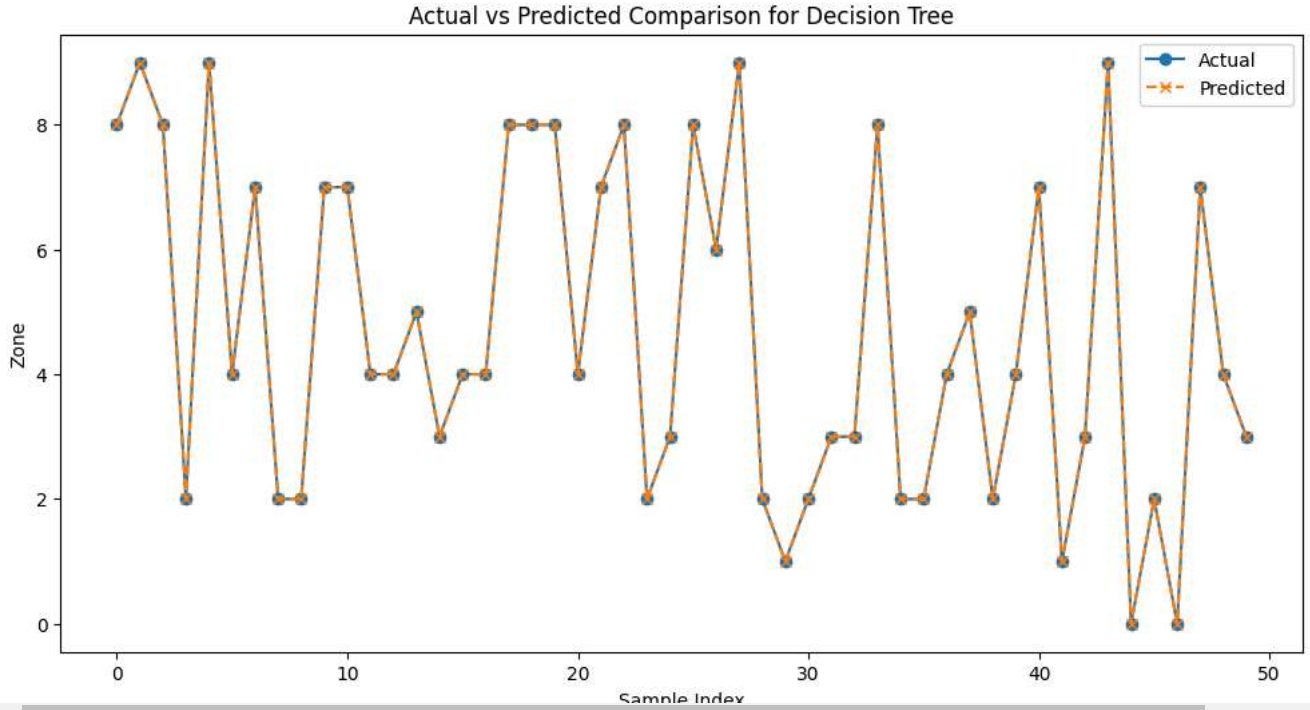



```
from sklearn.tree import DecisionTreeClassifier

# Initialize the Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(random_state=42)

# Fit the model on the training data
dt_classifier.fit(X_train, y_train)
# Predicting the test set results
y_pred_dt = dt_classifier.predict(X_test)



from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Calculate accuracy
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print(f"Decision Tree Model Accuracy: {accuracy_dt:.2f}")

# More detailed report
print("Classification Report for Decision Tree:")
print(classification_report(y_test, y_pred_dt))

# Confusion Matrix
conf_matrix_dt = confusion_matrix(y_test, y_pred_dt)
print("Confusion Matrix for Decision Tree:")
print(conf_matrix_dt)
```

```
Decision Tree Model Accuracy: 0.99
Classification Report for Decision Tree:
              precision    recall  f1-score   support
```

```
                0       0.99      0.99      0.99       649
                1       0.99      0.98      0.99       695
                2       1.00      1.00      1.00       896
                3       0.99      0.99      0.99       852
                4       0.99      0.99      0.99      1285
                5       0.99      0.99      0.99       530
                6       0.99      1.00      0.99       669
                7       0.99      0.98      0.99       656
                8       1.00      1.00      1.00       677
                9       0.99      1.00      0.99       805

         accuracy                           0.99      7714
        macro avg       0.99      0.99      0.99      7714
     weighted avg       0.99      0.99      0.99      7714

     Confusion Matrix for Decision Tree:
     [[ 645    0    2    0    0    0    0    0    0    2]
      [   0  683    0    3    2    2    5    0    0    0]
      [   1    0  893    0    0    2    0    0    0    0]
      [   0    1    0  847    0    3    1    0    0    0]
      [   0    2    0    0 1276    0    0    3    1    3]
      [   0    3    0    3    0  524    0    0    0    0]
      [   0    1    0    1    1    0  666    0    0    0]
      [   1    0    0    0    3    0    1  646    0    5]
      [   0    0    0    0    1    0    0    0  676    0]
      [   2    0    0    0    1    0    0    1    0  801]]
```

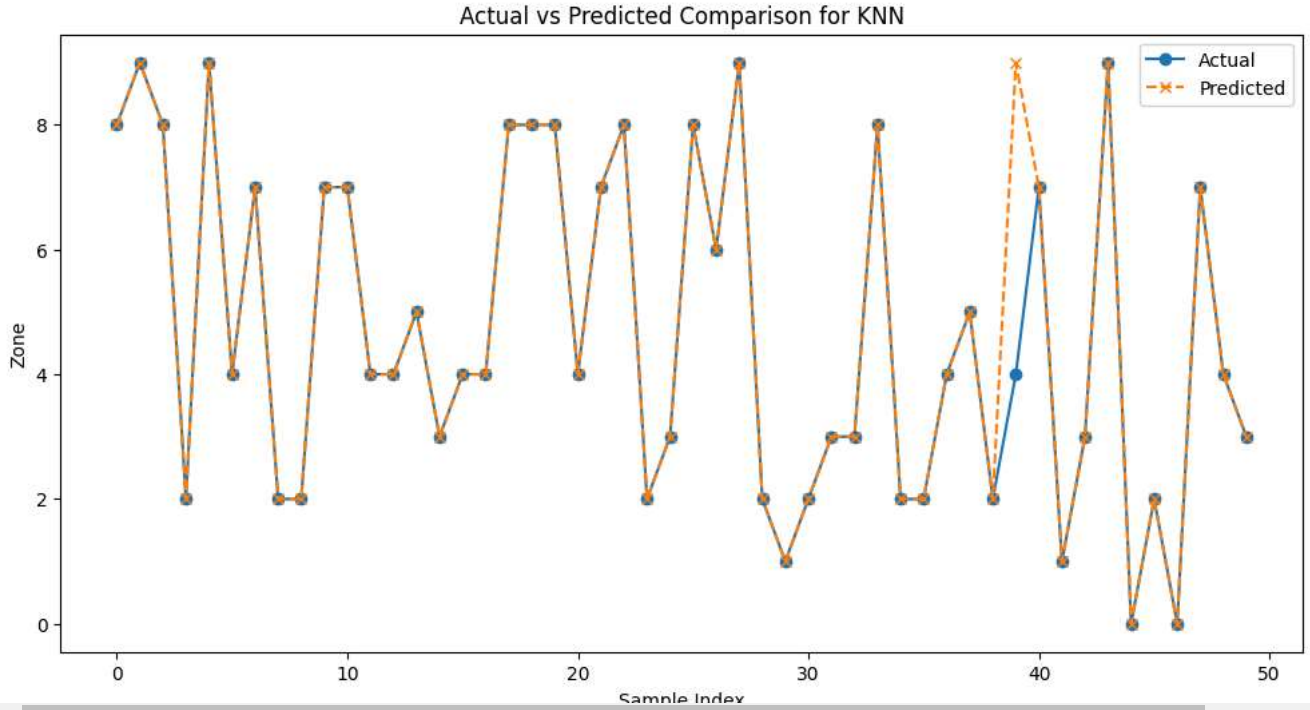
```
import matplotlib.pyplot as plt

# Plotting a sample of the results
plt.figure(figsize=(12, 6))
sample_results_dt = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_dt}).sample(50, random_state=42)
plt.plot(sample_results_dt['Actual'].values, label='Actual', marker='o')
plt.plot(sample_results_dt['Predicted'].values, label='Predicted', linestyle='--', marker='x')
plt.title('Actual vs Predicted Comparison for Decision Tree')
plt.ylabel('Zone')
plt.xlabel('Sample Index')
plt.legend()
plt.show()
```



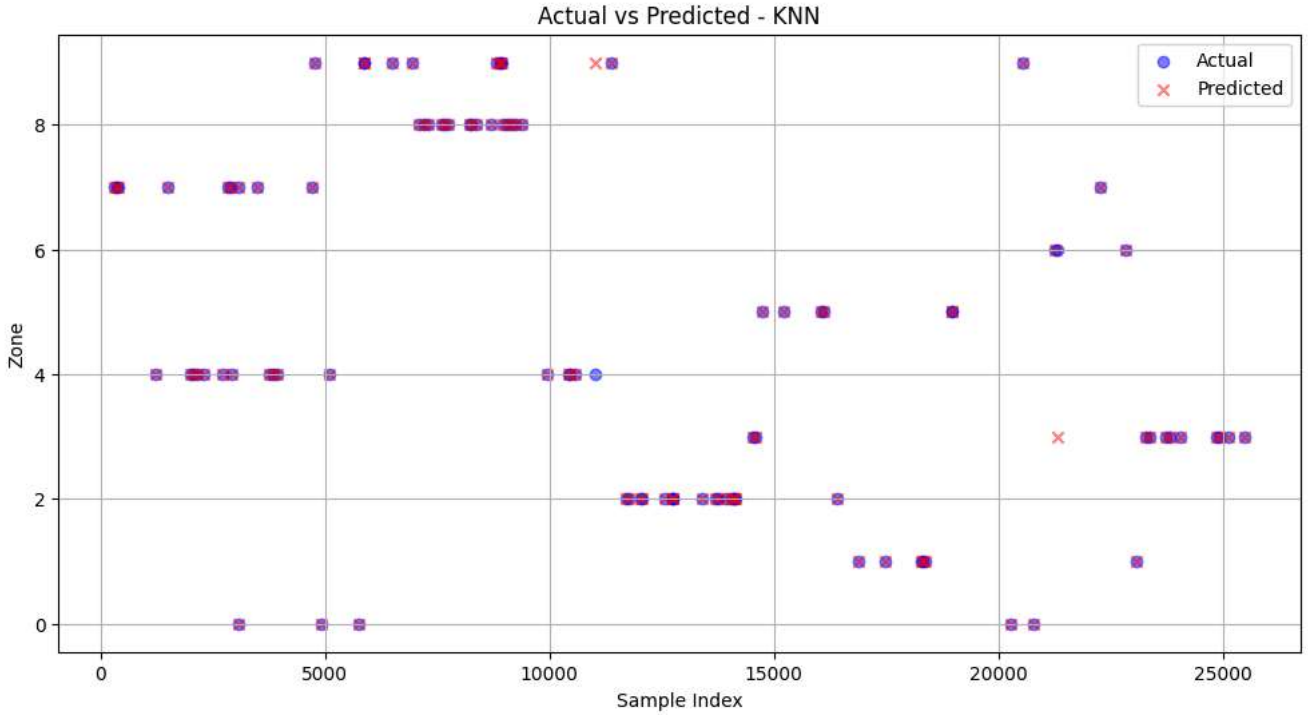

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
# Initialize the KNN model
# Starting with k=5, a common default; adjust based on performance
knn_classifier = KNeighborsClassifier(n_neighbors=5)

# Fit the model on the training data
knn_classifier.fit(X_train, y_train)
# Making predictions
y_pred_knn = knn_classifier.predict(X_test)



# Calculate accuracy
accuracy_knn = accuracy_score(y_test, y_pred_knn)
print(f"KNN Model Accuracy: {accuracy_knn:.2f}")

# Detailed classification report
print("Classification Report for KNN:")
print(classification_report(y_test, y_pred_knn))

# Confusion Matrix
conf_matrix_knn = confusion_matrix(y_test, y_pred_knn)
print("Confusion Matrix for KNN:")
print(conf_matrix_knn)
```

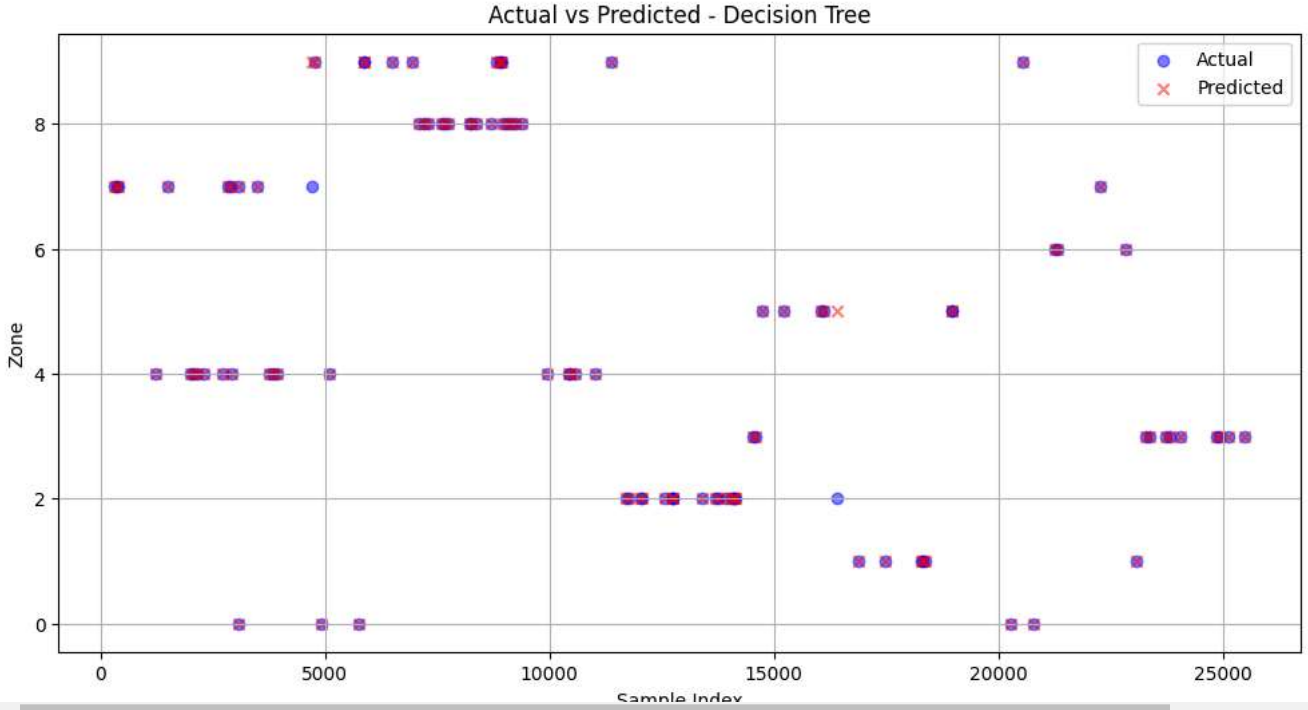
```
KNN Model Accuracy: 0.98
Classification Report for KNN:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       649
           1       0.96      0.97      0.96       695
           2       1.00      1.00      1.00       896
           3       0.98      0.99      0.99       852
           4       0.97      0.97      0.97      1285
           5       0.98      0.98      0.98       530
           6       0.98      0.98      0.98       669
           7       0.98      0.98      0.98       656
           8       0.98      0.99      0.98       677
           9       0.97      0.96      0.96       805

    accuracy                           0.98      7714
   macro avg       0.98      0.98      0.98      7714
weighted avg       0.98      0.98      0.98      7714

Confusion Matrix for KNN:
[[ 641    0    2    0    0    0    0    2    0    4]
 [   0  671    0    2    8    8    4    0    2    0]
 [   2    0  894    0    0    0    0    0    0    0]
 [   0    3    0  844    0    1    4    0    0    0]
 [   0   12    0    0 1250    0    1    5    6   11]
 [   0    4    0    4    0  522    0    0    0    0]
 [   0    7    0    7    1    0  653    1    0    0]
 [   0    0    0    0    7    0    1  640    0    8]
 [   0    0    0    0    5    0    0    0  669    3]
 [  12    0    0    0   12    0    0    3    7  771]]
```

```
import matplotlib.pyplot as plt

# Plotting a sample of the results
plt.figure(figsize=(12, 6))
sample_results_knn = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_knn}).sample(50, random_state=42)
plt.plot(sample_results_knn['Actual'].values, label='Actual', marker='o')
plt.plot(sample_results_knn['Predicted'].values, label='Predicted', linestyle='--', marker='x')
plt.title('Actual vs Predicted Comparison for KNN')
plt.ylabel('Zone')
plt.xlabel('Sample Index')
plt.legend()
plt.show()
```




```
import matplotlib.pyplot as plt

def plot_actual_vs_predicted(y_test, y_pred, model_name):
    sample_results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}).sample(100, random_state=42)  # sampling for visualization
    plt.figure(figsize=(12, 6))
    plt.scatter(sample_results.index, sample_results['Actual'], color='blue', label='Actual', alpha=0.5, marker='o')
    plt.scatter(sample_results.index, sample_results['Predicted'], color='red', label='Predicted', alpha=0.5, marker='x')
    plt.title(f'Actual vs Predicted - {model_name}')
    plt.xlabel('Sample Index')
    plt.ylabel('Zone')
    plt.legend()
    plt.grid(True)
    plt.show()

# Plot for KNN
plot_actual_vs_predicted(y_test, y_pred_knn, "KNN")

# Plot for Decision Tree
plot_actual_vs_predicted(y_test, y_pred_dt, "Decision Tree")
```


Actual vs Predicted - KNN
Actual
Predicted
Zone
Sample Index
0
2
4
6
8
0
5000
10000
15000
20000
25000


Actual vs Predicted - Decision Tree
Actual
Predicted
Zone
Sample Index
0
2
4
6
8
0
5000
10000
15000
20000
25000

```
# Concatenate along columns
merged_df = pd.concat([df_clean, bagow_df], axis=1)



Cmerged_df = pd.concat([merged_df, bagofw_df], axis=1)


# Dropping the 'LSOA name' and 'Last outcome category' columns
Cmerged_df = Cmerged_df.drop(['LSOA name', 'Last outcome category'], axis=1)



from sklearn.feature_extraction.text import CountVectorizer

# Define your custom list of stop words
custm_stop_words = ['and']

# Initialize the CountVectorizer with your custom stop words
vectorizer = CountVectorizer(stop_words=custm_stop_words)

# Fit and transform the data
X = vectorizer.fit_transform(Cmerged_df['Crime type'])

# Convert to DataFrame for better readability
bgow_df = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out())

print(bgow_df)
```

```
       arson  bicycle  burglary  crime  criminal  damage  drugs  from  of  \
0          0        0         0      1         0       0      0     0   0
1          0        0         0      0         0       0      0     0   0
2          0        0         0      0         0       0      0     0   0
3          0        0         0      0         0       0      0     0   0
4          0        0         0      0         0       0      0     0   0
...      ...      ...       ...    ...       ...     ...    ...   ...  ..
25708      0        0         0      0         0       0      0     0   0
25709      0        0         0      0         0       0      0     0   0
25710      0        0         0      0         0       0      0     0   0
25711      0        0         0      1         0       0      0     0   0
25712      0        0         0      0         0       0      0     0   0

       offences  ...  possession  public  robbery  sexual  shoplifting  the  \
0             0  ...           0       0        0       0            0    0
1             1  ...           0       0        0       1            0    0
2             1  ...           0       0        0       1            0    0
3             1  ...           0       0        0       1            0    0
4             1  ...           0       0        0       1            0    0
...         ...  ...         ...     ...      ...     ...          ...  ...
25708         1  ...           0       0        0       1            0    0
25709         0  ...           0       0        1       0            0    0
25710         0  ...           0       0        1       0            0    0
25711         0  ...           0       0        0       0            0    0
25712         1  ...           0       0        0       1            0    0

       theft  vehicle  violence  weapons
0          0        1         0        0
1          0        0         1        0
2          0        0         1        0
```