m189_updated

September 23, 2025

[4]:
```
!pip install kagglehub
```

```
Collecting plotly
  Downloading plotly-6.0.0-py3-none-any.whl.metadata (5.6 kB)
Collecting narwhals>=1.15.1 (from plotly)
  Downloading narwhals-1.30.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: packaging in
/opt/homebrew/Cellar/jupyterlab/4.3.4_1/libexec/lib/python3.13/site-packages
(from plotly) (24.2)
Downloading plotly-6.0.0-py3-none-any.whl (14.8 MB)
                         14.8/14.8 MB
39.4 MB/s eta 0:00:0000:010:01
Downloading narwhals-1.30.0-py3-none-any.whl (313 kB)
Installing collected packages: narwhals, plotly
Successfully installed narwhals-1.30.0 plotly-6.0.0

[notice] A new release of pip is
available: 24.3.1 -> 25.0.1
[notice] To update, run:
/opt/homebrew/Cellar/jupyterlab/4.3.4_1/libexec/bin/python -m pip

install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

[5]:
```python
import kagglehub
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import plotly.express as px
import seaborn as sns
```

[6]:
```python
weather_path = kagglehub.dataset_download("selfishgene/
↪historical-hourly-weather-data")
```

```python
wildfire_path = kagglehub.dataset_download("rtatman/188-million-us-wildfires")

print("Weather Data Path:", weather_path)
print("Wildfire Data Path:", wildfire_path)
```

```
Weather Data Path:
/Users/katelynwong/.cache/kagglehub/datasets/selfishgene/historical-hourly-
weather-data/versions/2
Wildfire Data Path:
/Users/katelynwong/.cache/kagglehub/datasets/rtatman/188-million-us-
wildfires/versions/2
```

[7]:
```python
import os

# List all files in the wildfire and weather dataset directories
wildfire_files = os.listdir(wildfire_path)
weather_files = os.listdir(weather_path)

print("Wildfire Dataset Files:", wildfire_files)
print("Weather Dataset Files:", weather_files)
```

```
Wildfire Dataset Files: ['FPA_FOD_20170508.sqlite']
Weather Dataset Files: ['weather_description.csv', 'humidity.csv',
'wind_direction.csv', 'temperature.csv', 'pressure.csv', 'city_attributes.csv',
'wind_speed.csv']
```

[8]:
```python
city_attributes_df = pd.read_csv(os.path.join(weather_path, 'city_attributes.
 ↪csv'))
#print(city_attributes_df.shape)
print(city_attributes_df.head())
#print(len(city_attributes_df['City'].unique()))
```

```
            City        Country   Latitude   Longitude
0      Vancouver         Canada  49.249660 -123.119339
1       Portland  United States  45.523449 -122.676208
2  San Francisco  United States  37.774929 -122.419418
3        Seattle  United States  47.606209 -122.332069
4    Los Angeles  United States  34.052231 -118.243683
```

[9]:
```python
weather_description_df = pd.read_csv(os.path.join(weather_path,␣
 ↪'weather_description.csv'))
weather_description_df = weather_description_df[1:]
print(weather_description_df.columns)
```

```
Index(['datetime', 'Vancouver', 'Portland', 'San Francisco', 'Seattle',
       'Los Angeles', 'San Diego', 'Las Vegas', 'Phoenix', 'Albuquerque',
       'Denver', 'San Antonio', 'Dallas', 'Houston', 'Kansas City',
       'Minneapolis', 'Saint Louis', 'Chicago', 'Nashville', 'Indianapolis',
       'Atlanta', 'Detroit', 'Jacksonville', 'Charlotte', 'Miami',
```

```
       'Pittsburgh', 'Toronto', 'Philadelphia', 'New York', 'Montreal',
       'Boston', 'Beersheba', 'Tel Aviv District', 'Eilat', 'Haifa',
       'Nahariyya', 'Jerusalem'],
      dtype='object')
```

[10]:
```python
# Load a specific weather file (e.g., temperature.csv)
temperature_df = pd.read_csv(os.path.join(weather_path, 'temperature.csv'))
temperature_df = temperature_df[1:]
print(temperature_df['datetime'].unique())
```

```
['2012-10-01 13:00:00' '2012-10-01 14:00:00' '2012-10-01 15:00:00' …
 '2017-11-29 22:00:00' '2017-11-29 23:00:00' '2017-11-30 00:00:00']
```

[11]:
```python
# Load a specific weather file (e.g., temperature.csv)
humidity_df = pd.read_csv(os.path.join(weather_path, 'humidity.csv'))
humidity_df = humidity_df[1:]
print(humidity_df.head())
print(humidity_df.shape)
```

```
              datetime  Vancouver  Portland  San Francisco  Seattle  \
1  2012-10-01 13:00:00       76.0      81.0           88.0     81.0
2  2012-10-01 14:00:00       76.0      80.0           87.0     80.0
3  2012-10-01 15:00:00       76.0      80.0           86.0     80.0
4  2012-10-01 16:00:00       77.0      80.0           85.0     79.0
5  2012-10-01 17:00:00       78.0      79.0           84.0     79.0

   Los Angeles  San Diego  Las Vegas  Phoenix  Albuquerque  …  Philadelphia  \
1         88.0       82.0       22.0     23.0         50.0  …          71.0
2         88.0       81.0       21.0     23.0         49.0  …          70.0
3         88.0       81.0       21.0     23.0         49.0  …          70.0
4         88.0       81.0       21.0     23.0         49.0  …          69.0
5         88.0       80.0       21.0     24.0         49.0  …          69.0

   New York  Montreal  Boston  Beersheba  Tel Aviv District  Eilat  Haifa  \
1      58.0      93.0    68.0       50.0               63.0   22.0   51.0
2      57.0      91.0    68.0       51.0               62.0   22.0   51.0
3      57.0      87.0    68.0       51.0               62.0   22.0   51.0
4      57.0      84.0    68.0       52.0               62.0   22.0   51.0
5      57.0      80.0    68.0       54.0               62.0   23.0   51.0

   Nahariyya  Jerusalem
1       51.0       50.0
2       51.0       50.0
3       51.0       50.0
4       51.0       50.0
5       51.0       50.0

[5 rows x 37 columns]
(45252, 37)
```

[12]:
```python
# Load a specific weather file (e.g., temperature.csv)
wind_direction_df = pd.read_csv(os.path.join(weather_path, 'wind_direction.
↪csv'))
wind_direction_df = wind_direction_df[1:]
wind_direction_df.head()
```

[12]:
```
              datetime  Vancouver  Portland  San Francisco  Seattle  \
1  2012-10-01 13:00:00        0.0       0.0          150.0      0.0
2  2012-10-01 14:00:00        6.0       4.0          147.0      2.0
3  2012-10-01 15:00:00       20.0      18.0          141.0     10.0
4  2012-10-01 16:00:00       34.0      31.0          135.0     17.0
5  2012-10-01 17:00:00       47.0      44.0          129.0     24.0

   Los Angeles  San Diego  Las Vegas  Phoenix  Albuquerque  …  Philadelphia  \
1          0.0        0.0        0.0     10.0        360.0  …         270.0
2          0.0        0.0        8.0      9.0        360.0  …         270.0
3          0.0        0.0       23.0      9.0        360.0  …         271.0
4          0.0        0.0       37.0      9.0        360.0  …         272.0
5          0.0        0.0       51.0      8.0        360.0  …         274.0

   New York  Montreal  Boston  Beersheba  Tel Aviv District  Eilat  Haifa  \
1     260.0     230.0    60.0      135.0              101.0   30.0  336.0
2     260.0     230.0    60.0      157.0              315.0   30.0  336.0
3     260.0     231.0    60.0      157.0              307.0   30.0  336.0
4     260.0     233.0    60.0      157.0              294.0   30.0  336.0
5     261.0     234.0    61.0      157.0              282.0   30.0  336.0

   Nahariyya  Jerusalem
1      336.0      329.0
2      336.0      329.0
3      336.0      329.0
4      336.0      329.0
5      336.0      329.0

[5 rows x 37 columns]
```

[13]:
```python
pressure_df = pd.read_csv(os.path.join(weather_path, 'pressure.csv'))
pressure_df = pressure_df[1:]

wind_speed_df = pd.read_csv(os.path.join(weather_path, 'wind_speed.csv'))
wind_speed_df = wind_speed_df[1:]
```

[14]:
```python
for df in [humidity_df, wind_direction_df, temperature_df, pressure_df,␣
↪wind_speed_df]:
    df.rename(columns={'datetime': 'Datetime'}, inplace=True)  # Ensure same␣
↪column name
    df['Datetime'] = pd.to_datetime(df['Datetime'])
```

```python
# Function to reshape the data
def reshape_df(df, variable_name):
    return df.melt(id_vars=['Datetime'], var_name='City',␣
 ↪value_name=variable_name)

# Reshape each DataFrame
humidity_melted = reshape_df(humidity_df, 'Humidity')
wind_direction_melted = reshape_df(wind_direction_df, 'Wind_Direction')
temperature_melted = reshape_df(temperature_df, 'Temperature')
pressure_melted = reshape_df(pressure_df, 'Pressure')
wind_speed_melted = reshape_df(wind_speed_df, 'Wind_Speed')
#print(wind_speed_melted)

# Merge all DataFrames on 'Datetime' and 'City'
weather_df = humidity_melted.merge(wind_direction_melted, on=['Datetime',␣
 ↪'City']) \
                           .merge(temperature_melted, on=['Datetime', 'City']) \
                           .merge(pressure_melted, on=['Datetime', 'City']) \
                           .merge(wind_speed_melted, on=['Datetime', 'City'])
print(weather_df['City'].value_counts())
print(weather_df['Wind_Speed'].value_counts())

# Drop 'Datetime' and compute correlation
#correlation_matrix = weather_df.drop(columns=['Datetime']).corr()
#print(correlation_matrix)

'''

# Plot the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f",␣
 ↪linewidths=0.5)
plt.title("Correlation Heatmap of Weather Parameters")
plt.show()
'''
```

```
City
Vancouver           45252
Portland            45252
Detroit             45252
Jacksonville        45252
Charlotte           45252
Miami               45252
Pittsburgh          45252
Toronto             45252
Philadelphia        45252
```

```
New York              45252
Montreal              45252
Boston                45252
Beersheba             45252
Tel Aviv District     45252
Eilat                 45252
Haifa                 45252
Nahariyya             45252
Atlanta               45252
Indianapolis          45252
Nashville             45252
Albuquerque           45252
San Francisco         45252
Seattle               45252
Los Angeles           45252
San Diego             45252
Las Vegas             45252
Phoenix               45252
Denver                45252
Chicago               45252
San Antonio           45252
Dallas                45252
Houston               45252
Kansas City           45252
Minneapolis           45252
Saint Louis           45252
Jerusalem             45252
Name: count, dtype: int64
Wind_Speed
1.0     396311
2.0     351061
3.0     267348
4.0     191482
5.0     125224
0.0     118887
6.0      75656
7.0      43707
8.0      24085
9.0      12954
10.0      6743
11.0      3509
12.0      1912
13.0      1070
14.0       501
15.0       279
17.0       116
16.0       110
18.0        49
```

```
19.0         38
20.0         20
21.0         11
22.0          8
23.0          5
25.0          4
27.0          4
28.0          3
24.0          2
49.0          2
41.0          2
43.0          2
35.0          2
36.0          1
31.0          1
50.0          1
34.0          1
48.0          1
30.0          1
44.0          1
Name: count, dtype: int64
```

[14]:
```
'\n\n# Plot the heatmap\nplt.figure(figsize=(10,
6))\nsns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f",
linewidths=0.5)\nplt.title("Correlation Heatmap of Weather
Parameters")\nplt.show()\n'
```

[15]:
```
import sqlite3
import pandas as pd
import os

wildfire_db_path = os.path.join(wildfire_path, 'FPA_FOD_20170508.sqlite')
conn = sqlite3.connect(wildfire_db_path)

fire_data = pd.read_sql("SELECT * FROM Fires", conn)
# Close the connection when done
conn.close()
print(fire_data.head())

# List all tables in the database
#query = "SELECT name FROM sqlite_master WHERE type='table';"
#tables = pd.read_sql(query, conn)
#print("Tables in the database:", tables)
```

```
   OBJECTID  FOD_ID      FPA_ID SOURCE_SYSTEM_TYPE SOURCE_SYSTEM  \
0         1       1  FS-1418826                FED   FS-FIRESTAT
1         2       2  FS-1418827                FED   FS-FIRESTAT
2         3       3  FS-1418835                FED   FS-FIRESTAT
```

```
3         4       4  FS-1418845                FED   FS-FIRESTAT
4         5       5  FS-1418847                FED   FS-FIRESTAT

  NWCG_REPORTING_AGENCY NWCG_REPORTING_UNIT_ID  NWCG_REPORTING_UNIT_NAME  \
0                    FS                USCAPNF    Plumas National Forest
1                    FS                USCAENF  Eldorado National Forest
2                    FS                USCAENF  Eldorado National Forest
3                    FS                USCAENF  Eldorado National Forest
4                    FS                USCAENF  Eldorado National Forest

  SOURCE_REPORTING_UNIT SOURCE_REPORTING_UNIT_NAME  … FIRE_SIZE_CLASS  \
0                  0511     Plumas National Forest  …               A
1                  0503   Eldorado National Forest  …               A
2                  0503   Eldorado National Forest  …               A
3                  0503   Eldorado National Forest  …               A
4                  0503   Eldorado National Forest  …               A

    LATITUDE   LONGITUDE OWNER_CODE       OWNER_DESCR STATE COUNTY FIPS_CODE  \
0  40.036944 -121.005833        5.0              USFS    CA     63       063
1  38.933056 -120.404444        5.0              USFS    CA     61       061
2  38.984167 -120.735556       13.0  STATE OR PRIVATE    CA     17       017
3  38.559167 -119.913333        5.0              USFS    CA      3       003
4  38.559167 -119.933056        5.0              USFS    CA      3       003

   FIPS_NAME                                              Shape
0     Plumas  b'\x00\x01\xad\x10\x00\x00\xe8d\xc2\x92_@^\xc0…
1     Placer  b'\x00\x01\xad\x10\x00\x00T\xb6\xeej\xe2\x19^\…
2  El Dorado  b'\x00\x01\xad\x10\x00\x00\xd0\xa5\xa0W\x13/^\…
3     Alpine  b'\x00\x01\xad\x10\x00\x00\x94\xac\xa3\rt\xfa]…
4     Alpine  b'\x00\x01\xad\x10\x00\x00@\xe3\xaa.\xb7\xfb]\…

[5 rows x 39 columns]
```

0.1 heads up, what's below takes time to run

[18]:
```
print(fire_data.head())
print(fire_data.info())
fire_data.shape
```

```
   OBJECTID  FOD_ID      FPA_ID SOURCE_SYSTEM_TYPE SOURCE_SYSTEM  \
0         1       1  FS-1418826                FED   FS-FIRESTAT
1         2       2  FS-1418827                FED   FS-FIRESTAT
2         3       3  FS-1418835                FED   FS-FIRESTAT
3         4       4  FS-1418845                FED   FS-FIRESTAT
4         5       5  FS-1418847                FED   FS-FIRESTAT

  NWCG_REPORTING_AGENCY NWCG_REPORTING_UNIT_ID  NWCG_REPORTING_UNIT_NAME  \
0                    FS                USCAPNF    Plumas National Forest
```

```
1                    FS                USCAENF  Eldorado National Forest
2                    FS                USCAENF  Eldorado National Forest
3                    FS                USCAENF  Eldorado National Forest
4                    FS                USCAENF  Eldorado National Forest

  SOURCE_REPORTING_UNIT SOURCE_REPORTING_UNIT_NAME  … FIRE_SIZE_CLASS  \
0                  0511     Plumas National Forest  …               A
1                  0503   Eldorado National Forest  …               A
2                  0503   Eldorado National Forest  …               A
3                  0503   Eldorado National Forest  …               A
4                  0503   Eldorado National Forest  …               A

    LATITUDE   LONGITUDE OWNER_CODE       OWNER_DESCR STATE COUNTY FIPS_CODE  \
0  40.036944 -121.005833        5.0              USFS    CA     63       063
1  38.933056 -120.404444        5.0              USFS    CA     61       061
2  38.984167 -120.735556       13.0  STATE OR PRIVATE    CA     17       017
3  38.559167 -119.913333        5.0              USFS    CA      3       003
4  38.559167 -119.933056        5.0              USFS    CA      3       003

   FIPS_NAME                                              Shape
0     Plumas  b'\x00\x01\xad\x10\x00\x00\xe8d\xc2\x92_@^\xc0…
1     Placer  b'\x00\x01\xad\x10\x00\x00T\xb6\xeej\xe2\x19^\…
2  El Dorado  b'\x00\x01\xad\x10\x00\x00\xd0\xa5\xa0W\x13/^\…
3     Alpine  b'\x00\x01\xad\x10\x00\x00\x94\xac\xa3\rt\xfa]…
4     Alpine  b'\x00\x01\xad\x10\x00\x00@\xe3\xaa.\xb7\xfb]\…

[5 rows x 39 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1880465 entries, 0 to 1880464
Data columns (total 39 columns):
 #   Column                      Dtype
---  ------                      -----
 0   OBJECTID                    int64
 1   FOD_ID                      int64
 2   FPA_ID                      object
 3   SOURCE_SYSTEM_TYPE          object
 4   SOURCE_SYSTEM               object
 5   NWCG_REPORTING_AGENCY       object
 6   NWCG_REPORTING_UNIT_ID      object
 7   NWCG_REPORTING_UNIT_NAME    object
 8   SOURCE_REPORTING_UNIT       object
 9   SOURCE_REPORTING_UNIT_NAME  object
 10  LOCAL_FIRE_REPORT_ID        object
 11  LOCAL_INCIDENT_ID           object
 12  FIRE_CODE                   object
 13  FIRE_NAME                   object
 14  ICS_209_INCIDENT_NUMBER     object
 15  ICS_209_NAME                object
```

```
 16  MTBS_ID                     object
 17  MTBS_FIRE_NAME              object
 18  COMPLEX_NAME                object
 19  FIRE_YEAR                   int64
 20  DISCOVERY_DATE              float64
 21  DISCOVERY_DOY               int64
 22  DISCOVERY_TIME              object
 23  STAT_CAUSE_CODE             float64
 24  STAT_CAUSE_DESCR            object
 25  CONT_DATE                   float64
 26  CONT_DOY                    float64
 27  CONT_TIME                   object
 28  FIRE_SIZE                   float64
 29  FIRE_SIZE_CLASS             object
 30  LATITUDE                    float64
 31  LONGITUDE                   float64
 32  OWNER_CODE                  float64
 33  OWNER_DESCR                 object
 34  STATE                       object
 35  COUNTY                      object
 36  FIPS_CODE                   object
 37  FIPS_NAME                   object
 38  Shape                       object
dtypes: float64(8), int64(4), object(27)
memory usage: 559.5+ MB
None
```

[18]: (1880465, 39)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1880465 entries, 0 to 1880464
Data columns (total 39 columns):
 #   Column                      Dtype
---  ------                      -----
 0   OBJECTID                    int64
 1   FOD_ID                      int64
 2   FPA_ID                      object
 3   SOURCE_SYSTEM_TYPE          object
 4   SOURCE_SYSTEM               object
 5   NWCG_REPORTING_AGENCY       object
 6   NWCG_REPORTING_UNIT_ID      object
 7   NWCG_REPORTING_UNIT_NAME    object
 8   SOURCE_REPORTING_UNIT       object
 9   SOURCE_REPORTING_UNIT_NAME  object
 10  LOCAL_FIRE_REPORT_ID        object
 11  LOCAL_INCIDENT_ID           object
 12  FIRE_CODE                   object
 13  FIRE_NAME                   object
```

```
 14   ICS_209_INCIDENT_NUMBER     object
 15   ICS_209_NAME                object
 16   MTBS_ID                     object
 17   MTBS_FIRE_NAME              object
 18   COMPLEX_NAME                object
 19   FIRE_YEAR                   int64
 20   DISCOVERY_DATE              float64
 21   DISCOVERY_DOY               int64
 22   DISCOVERY_TIME              object
 23   STAT_CAUSE_CODE             float64
 24   STAT_CAUSE_DESCR            object
 25   CONT_DATE                   float64
 26   CONT_DOY                    float64
 27   CONT_TIME                   object
 28   FIRE_SIZE                   float64
 29   FIRE_SIZE_CLASS             object
 30   LATITUDE                    float64
 31   LONGITUDE                   float64
 32   OWNER_CODE                  float64
 33   OWNER_DESCR                 object
 34   STATE                       object
 35   COUNTY                      object
 36   FIPS_CODE                   object
 37   FIPS_NAME                   object
 38   Shape                       object
dtypes: float64(8), int64(4), object(27)
memory usage: 559.5+ MB
None
```

0.2 randomly sampling to get a small subset for testing:

[19]:
```python
import pandas as pd

# Randomly sample 1800 rows from fire_data
fire_subset = fire_data.sample(n=20000, random_state=42)  # Set random_state␣
↪for reproducibility

# Reset index if needed
fire_subset = fire_subset.reset_index(drop=True)
```

[20]:
```python
fire_subset
```

[20]:
```
      OBJECTID      FOD_ID                          FPA_ID SOURCE_SYSTEM_TYPE  \
0      1644478   201772529                        W-663496                FED
1       156486      158040                       FS-385721                FED
2       440709      474869  SFO-GA01150606-40-029-0014-11             NONFED
3      1285274     1755089                SFO-WA-2008-5864             NONFED
```

4	417313	451251	SFO-GA00060503-36-091-0008-10	NONFED
…	…	…	…	…
19995	977466	1106128	TFS_FL_60542	NONFED
19996	1256086	1659323	SFO-GA-FY2002-Laurens-237	NONFED
19997	878592	1001182	SWRA_GA_57046	NONFED
19998	883192	1005859	SWRA_GA_61725	NONFED
19999	1732067	300017359	W-670524	FED

	SOURCE_SYSTEM	NWCG_REPORTING_AGENCY	NWCG_REPORTING_UNIT_ID	\
0	DOI-WFMI	NPS	USCAYNP	
1	FS-FIRESTAT	FS	USNMGNF	
2	ST-NASF	ST/C&L	USGAGAS	
3	ST-NASF	ST/C&L	USWAWAS	
4	ST-NASF	ST/C&L	USGAGAS	
…	…	…	…	
19995	ST-FLFLS	ST/C&L	USFLFLS	
19996	ST-NASF	ST/C&L	USGAGAS	
19997	ST-GAGAS	ST/C&L	USGAGAS	
19998	ST-GAGAS	ST/C&L	USGAGAS	
19999	DOI-WFMI	BIA	USMTFPA	

	NWCG_REPORTING_UNIT_NAME	SOURCE_REPORTING_UNIT	\
0	Yosemite National Park	CAYNP	
1	Gila National Forest	0306	
2	Georgia Forestry Commission	GA Statesboro	
3	Washington State Headquarters	WADNR	
4	Georgia Forestry Commission	GA McRae	
…	…	…	
19995	Florida Forest Service	FLFLS8	
19996	Georgia Forestry Commission	GAGAS	
19997	Georgia Forestry Commission	GAGAS27	
19998	Georgia Forestry Commission	GAGAS31	
19999	Fort Peck Agency	MTFPA	

	SOURCE_REPORTING_UNIT_NAME	…	FIRE_SIZE_CLASS	\
0	YOSEMITE NATIONAL PARK	…	A	
1	Gila National Forest	…	A	
2	GAS Ogeechee District, Statesboro Office	…	B	
3	Washington Department of Natural Resources	…	A	
4	GAS Ogeechee District, McRae Office	…	A	
…	…	…	…	
19995	FLS Waccasassa Forestry Center	…	B	
19996	Georgia Forestry Commission	…	C	
19997	GAS Unit 27	…	B	
19998	GAS Unit 31	…	A	
19999	Fort Peck Agency	…	A	

```
        LATITUDE   LONGITUDE OWNER_CODE            OWNER_DESCR STATE  \
0      37.516010 -119.623040        3.0                    NPS    CA
1      33.050833 -108.452500        5.0                   USFS    NM
2      32.172945  -81.464159       14.0  MISSING/NOT SPECIFIED    GA
3      48.059640 -123.186530        8.0                PRIVATE    WA
4      32.241889  -83.045710       14.0  MISSING/NOT SPECIFIED    GA
…            …          …          …                    …   …
19995  29.610000  -82.020000       14.0  MISSING/NOT SPECIFIED    FL
19996  32.629700  -83.068000        8.0                PRIVATE    GA
19997  31.927500  -84.640000       14.0  MISSING/NOT SPECIFIED    GA
19998  31.858300  -83.162500       14.0  MISSING/NOT SPECIFIED    GA
19999  48.114200 -105.174400        2.0                    BIA    MT

            COUNTY FIPS_CODE FIPS_NAME  \
0             None      None      None
1             None      None      None
2      Bryan North       029     Bryan
3        Clallam C       009   Clallam
4            Dodge       091     Dodge
…              …         …        …
19995       Putnam       107    Putnam
19996      Laurens       175   Laurens
19997         None      None      None
19998         None      None      None
19999         None      None      None

                                                   Shape
0      b'\x00\x01\xad\x10\x00\x00`\x06*\xe3\xdf\xe7]\…
1      b'\x00\x01\xad\x10\x00\x00(\\\x8f\xc2\xf5\x1c[…
2      b'\x00\x01\xad\x10\x00\x00\x889R\xc7\xb4]T\xc0…
3      b'\x00\x01\xad\x10\x00\x00@n\x86\x1b\xf0\xcb^\…
4      b'\x00\x01\xad\x10\x00\x00\x8c\xa2\xef\xea\xec…
…                                                    …
19995  b'\x00\x01\xad\x10\x00\x00\xe0z\x14\xaeG\x81T\…
19996  b'\x00\x01\xad\x10\x00\x000\x08\xac\x1cZ\xc4T\…
19997  b'\x00\x01\xad\x10\x00\x00(\\\x8f\xc2\xf5(U\xc…
19998  b'\x00\x01\xad\x10\x00\x00dffff\xcaT\xc00(~\x8…
19999  b'\x00\x01\xad\x10\x00\x00\x08\x1b\x9e^)KZ\xc0…

[20000 rows x 39 columns]
```

[18]:
```
!pip install geopy
```

```
Requirement already satisfied: geopy in c:\users\bhagya\appdata\local\packages\p
ythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-
packages\python310\site-packages (2.4.1)
Requirement already satisfied: geographiclib<3,>=1.52 in c:\users\bhagya\appdata
\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\lo
```

```
cal-packages\python310\site-packages (from geopy) (2.0)

[notice] A new release of pip is available: 23.0.1 -> 25.0.1
[notice] To update, run: C:\Users\Bhagya\AppData\Local\Microsoft\WindowsApps\Pyt
honSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\python.exe -m pip install
--upgrade pip
```

[21]:
```
(city_attributes_df).head()
```

[21]:

	City	Country	Latitude	Longitude
0	Vancouver	Canada	49.249660	-123.119339
1	Portland	United States	45.523449	-122.676208
2	San Francisco	United States	37.774929	-122.419418
3	Seattle	United States	47.606209	-122.332069
4	Los Angeles	United States	34.052231	-118.243683

[22]:
```
import pandas as pd
import numpy as np
from geopy.distance import great_circle

# Function to compute Haversine distance
def haversine(lat1, lon1, lat2, lon2):
    return great_circle((lat1, lon1), (lat2, lon2)).miles  # Distance in miles


# Define a threshold distance (in miles)
THRESHOLD_MILES = 100

# Create a list to store matches
matched_fires = []

# Compare each fire location with each city
for _, fire in fire_subset.iterrows():
    fire_lat, fire_lon = fire['LATITUDE'], fire['LONGITUDE']

    for _, city in city_attributes_df.iterrows():
        city_lat, city_lon = city['Latitude'], city['Longitude']
        distance = haversine(fire_lat, fire_lon, city_lat, city_lon)

        if distance <= THRESHOLD_MILES:
            matched_fires.append({
                'Fire_Lat': fire_lat,
                'Fire_Lon': fire_lon,
                'City': city['City'],
                'City_Lat': city_lat,
                'City_Lon': city_lon,
                'Distance_Miles': distance,
```

```
                "OBJECTID" : fire["OBJECTID"]
            })

# Convert to DataFrame
matched_fires_df = pd.DataFrame(matched_fires)

closest_cities_df = matched_fires_df.loc[
    matched_fires_df.groupby(['Fire_Lat', 'Fire_Lon'])['Distance_Miles'].
 ↪idxmin()
]
closest_cities_df
```

[22]:

```
       Fire_Lat    Fire_Lon       City   City_Lat    City_Lon  Distance_Miles  \
438   25.399054  -80.572002      Miami  25.774269  -80.193657       35.042800
5187  25.403080  -80.561420      Miami  25.774269  -80.193657       34.394381
3493  25.423600  -80.451000      Miami  25.774269  -80.193657       29.054640
971   25.425000  -80.549720      Miami  25.774269  -80.193657       32.781170
453   25.430000  -80.440000      Miami  25.774269  -80.193657       28.309274
…           …          …          …         …          …               …
3898  48.620000 -121.053000    Seattle  47.606209 -122.332069       91.583526
5864  48.852640 -122.280140  Vancouver  49.249660 -123.119339       46.867248
3435  48.887778 -121.612778  Vancouver  49.249660 -123.119339       72.634775
6048  48.902500 -121.691944  Vancouver  49.249660 -123.119339       68.911828
6230  48.969819 -121.946841  Vancouver  49.249660 -123.119339       56.445486

      OBJECTID
438     246029
5187   1798303
3493   1596109
971     416088
453     978504
…          …
3898   1736126
5864    763634
3435   1727072
6048     39662
6230    760273

[6176 rows x 7 columns]
```

[23]:

```
fire_subset_city = fire_subset[fire_subset['OBJECTID'].
 ↪isin(closest_cities_df['OBJECTID'])]
merged_df = pd.merge(fire_subset_city, closest_cities_df, on='OBJECTID',␣
 ↪how='inner')
merged_df = merged_df.reset_index(drop=True)
```

0.2.1 now we get the df that contains fire that is related to cities we have in the weather data

[24]:
```
merged_df["DISCOVERY_DOY"]
```

[24]:
```
0       261
1       183
2       111
3       278
4       195
       …
6171    257
6172    275
6173    240
6174    115
6175    328
Name: DISCOVERY_DOY, Length: 6176, dtype: int64
```

[25]:
```
merged_df.columns
```

[25]:
```
Index(['OBJECTID', 'FOD_ID', 'FPA_ID', 'SOURCE_SYSTEM_TYPE', 'SOURCE_SYSTEM',
       'NWCG_REPORTING_AGENCY', 'NWCG_REPORTING_UNIT_ID',
       'NWCG_REPORTING_UNIT_NAME', 'SOURCE_REPORTING_UNIT',
       'SOURCE_REPORTING_UNIT_NAME', 'LOCAL_FIRE_REPORT_ID',
       'LOCAL_INCIDENT_ID', 'FIRE_CODE', 'FIRE_NAME',
       'ICS_209_INCIDENT_NUMBER', 'ICS_209_NAME', 'MTBS_ID', 'MTBS_FIRE_NAME',
       'COMPLEX_NAME', 'FIRE_YEAR', 'DISCOVERY_DATE', 'DISCOVERY_DOY',
       'DISCOVERY_TIME', 'STAT_CAUSE_CODE', 'STAT_CAUSE_DESCR', 'CONT_DATE',
       'CONT_DOY', 'CONT_TIME', 'FIRE_SIZE', 'FIRE_SIZE_CLASS', 'LATITUDE',
       'LONGITUDE', 'OWNER_CODE', 'OWNER_DESCR', 'STATE', 'COUNTY',
       'FIPS_CODE', 'FIPS_NAME', 'Shape', 'Fire_Lat', 'Fire_Lon', 'City',
       'City_Lat', 'City_Lon', 'Distance_Miles'],
      dtype='object')
```

[26]:
```
weather_df = weather_df.dropna()
weather_df
```

[26]:
```
                    Datetime       City  Humidity  Wind_Direction  Temperature  \
20       2012-10-02 09:00:00  Vancouver      87.0           268.0   284.590217
21       2012-10-02 10:00:00  Vancouver      88.0           281.0   284.588174
22       2012-10-02 11:00:00  Vancouver      89.0           295.0   284.586130
23       2012-10-02 12:00:00  Vancouver      89.0           309.0   284.584087
24       2012-10-02 13:00:00  Vancouver      90.0           323.0   284.582043
…                        …          …        …               …            …
1628275  2017-10-27 20:00:00  Jerusalem      68.0             0.0   295.760000
1628276  2017-10-27 21:00:00  Jerusalem      72.0             0.0   293.150000
1628277  2017-10-27 22:00:00  Jerusalem      60.0             0.0   294.150000
1628278  2017-10-27 23:00:00  Jerusalem      56.0           150.0   294.150000
```

```
1628279 2017-10-28 00:00:00  Jerusalem      60.0             0.0   294.150000

         Pressure  Wind_Speed
20          807.0         0.0
21          849.0         0.0
22          890.0         0.0
23          932.0         0.0
24          973.0         0.0
…             …           …
1628275    1011.0         1.0
1628276    1011.0         1.0
1628277    1011.0         1.0
1628278    1011.0         3.0
1628279    1011.0         1.0

[1596318 rows x 7 columns]
```

[27]:
```python
merged_df['discovery_date'] = pd.to_datetime(merged_df['FIRE_YEAR'].astype(str)␣
↪+ '-' + merged_df['DISCOVERY_DOY'].astype(str), format='%Y-%j')
merged_df
```

[27]:
```
      OBJECTID     FOD_ID                        FPA_ID SOURCE_SYSTEM_TYPE  \
0      1285274    1755089              SFO-WA-2008-5864             NONFED
1       960838    1089097                   TFS_FL_35684             NONFED
2       687709     765280                    ME_01210080             NONFED
3      1602737  201623404        SFO-WI-2012-82120712012             NONFED
4       739585     838845                       SC_16565             NONFED
…         …          …                           …                  …
6171   1133610    1382295   CDF_2003_55_2225_070871                 NONFED
6172   1029303    1158945                  TFS_NC_207949             NONFED
6173   1510013  201193827           2011CACDFRRU079264             NONFED
6174    556716     598566        SFO-NJ0410-07_B042503             NONFED
6175    977466    1106128                   TFS_FL_60542             NONFED

     SOURCE_SYSTEM NWCG_REPORTING_AGENCY NWCG_REPORTING_UNIT_ID  \
0          ST-NASF                ST/C&L                USWAWAS
1         ST-FLFLS                ST/C&L                USFLFLS
2         ST-MEMES                ST/C&L                USMEMES
3          ST-NASF                ST/C&L                USWIWIS
4         ST-SCSCS                ST/C&L                USSCSCS
…             …                     …                      …
6171      ST-CACDF                ST/C&L                USCARRU
6172      ST-NCNCS                ST/C&L                USNCNCS
6173       ST-NASF                ST/C&L                USCARRU
6174       ST-NASF                ST/C&L                USNJNJS
6175      ST-FLFLS                ST/C&L                USFLFLS
```

```
                       NWCG_REPORTING_UNIT_NAME SOURCE_REPORTING_UNIT  \
0             Washington State Headquarters                 WADNR
1                     Florida Forest Service               FLFLS16
2                       Maine Forest Service                 MEMES
3     Wisconsin Department of Natural Resources              WIDNR
4            South Carolina Forestry Commission              SCSCS
…                                          …                   …
6171                           Riverside Unit                 CARRU
6172             North Carolina Forest Service             NCNCS210
6173                           Riverside Unit                 CARRU
6174            New Jersey Forest Fire Service                NJNJB
6175                    Florida Forest Service               FLFLS8

                      SOURCE_REPORTING_UNIT_NAME  … FIPS_CODE  FIPS_NAME  \
0     Washington Department of Natural Resources  …       009    Clallam
1                        FLS Okeechobee District  …       085     Martin
2                           Maine Forest Service  …       031       York
3      Wisconsin Department of Natural Resources  …       095       Polk
4             South Carolina Forestry Commission  …       059    Laurens
…                                            …  …       …        …
6171                       CDF - Riverside Unit  …      None       None
6172                     NCS Region 2 District 10  …      None       None
6173                       CDF - Riverside Unit  …       065  Riverside
6174   New Jersey Forest Fire Service Division B  …       029      Ocean
6175                 FLS Waccasassa Forestry Center  …       107     Putnam

                                                   Shape   Fire_Lat  \
0     b'\x00\x01\xad\x10\x00\x00@n\x86\x1b\xf0\xcb^\…  48.059640
1     b'\x00\x01\xad\x10\x00\x00x\x14\xaeG\xe1\nT\xc…  27.060000
2     b"\x00\x01\xad\x10\x00\x00d'\xbdN\xae\xacQ\xc0…  43.443194
3     b'\x00\x01\xad\x10\x00\x00\xf81\xe6\xae%\x16W\…  45.704740
4     b'\x00\x01\xad\x10\x00\x00\x98\x99\x99\x99\x99…  34.595833
…                                                  …        …
6171  b'\x00\x01\xad\x10\x00\x00|\x12~-\xd8V]\xc0X\x…  33.996944
6172  b"\x00\x01\xad\x10\x00\x00\xe8Q\xb8\x1e\x853T\…  36.191700
6173  b'\x00\x01\xad\x10\x00\x00\xa4o\x99\xd3eF]\xc0…  33.602267
6174  b'\x00\x01\xad\x10\x00\x00\xfc\xd4x\xe9&\x8dR\…  39.958500
6175  b'\x00\x01\xad\x10\x00\x00\xe0z\x14\xaeG\x81T\…  29.610000

        Fire_Lon          City   City_Lat    City_Lon Distance_Miles  \
0    -123.186530       Seattle  47.606209 -122.332069      50.518575
1     -80.170000         Miami  25.774269  -80.193657      88.847614
2     -70.698139        Boston  42.358429  -71.059769      77.152440
3     -92.346050   Minneapolis  44.979969  -93.263840      67.038499
4     -82.087500     Charlotte  35.227089  -80.843132      82.903997
…            …             …          …           …            …
6171 -117.356944   Los Angeles  34.052231 -118.243683      50.921918
```

```
6172  -80.805000     Charlotte  35.227089  -80.843132      66.682598
6173 -117.099965     San Diego  32.715328 -117.157257      61.371182
6174  -74.205500  Philadelphia  39.952339  -75.163788      50.755513
6175  -82.020000  Jacksonville  30.332180  -81.655647      54.455263

      discovery_date
0         2008-09-17
1         1998-07-02
2         2001-04-21
3         2012-10-04
4         1993-07-14
…                …
6171      2003-09-14
6172      1998-10-02
6173      2011-08-28
6174      2007-04-25
6175      1997-11-24

[6176 rows x 46 columns]
```

[28]:

```python
import pandas as pd

# Assuming merged_df and weather_df are your existing DataFrames

# Convert discovery_date and Datetime to datetime objects if they aren't already
merged_df['discovery_date'] = pd.to_datetime(merged_df['discovery_date'])
weather_df['Datetime'] = pd.to_datetime(weather_df['Datetime'])

# Initialize an empty list to store the matched rows
matched_rows = []

# Set time window for comparison
time_window = pd.Timedelta(hours=12)

# Iterate over each row in merged_df
for _, merged_row in merged_df.iterrows():
    # Get the corresponding city and time from merged_df
    city = merged_row['City']
    discovery_time = merged_row['discovery_date']

    # Find rows in weather_df with the same city and a matching time within the␣
 ↪time window
    potential_matches = weather_df[
        (weather_df['City'] == city) &
        (abs(weather_df['Datetime'] - discovery_time) <= time_window)
    ]
```

```
    # If there are matches, append to the list
    if not potential_matches.empty:
        for _, weather_row in potential_matches.iterrows():
            # Combine both merged_df and weather_df rows into a DataFrame
            combined_row = pd.concat([merged_row, weather_row], axis=0)
            matched_rows.append(combined_row.to_frame().T)  # Convert to␣
↪DataFrame and append
```

0.3 we want to relate the weather data and fire data, here is what I did:

0.3.1 weather data and fire data both have time and location, so I merge them if the time and location are same

0.3.2 now we have a final_df that contains fire data, and related weather data at the same time and location.

[29]:
```
# Create a new DataFrame from the list of matched rows
final_df = pd.concat(matched_rows, ignore_index=True)
final_df
```

[29]:
```
       OBJECTID     FOD_ID                           FPA_ID SOURCE_SYSTEM_TYPE  \
0       1602737  201623404        SFO-WI-2012-82120712012             NONFED
1       1602737  201623404        SFO-WI-2012-82120712012             NONFED
2       1602737  201623404        SFO-WI-2012-82120712012             NONFED
3       1602737  201623404        SFO-WI-2012-82120712012             NONFED
4       1602737  201623404        SFO-WI-2012-82120712012             NONFED
…           …          …                            …                  …
19487   1749796  300090789  SFO-2014GAGAS-FY2014-Dade-025             NONFED
19488   1749796  300090789  SFO-2014GAGAS-FY2014-Dade-025             NONFED
19489   1749796  300090789  SFO-2014GAGAS-FY2014-Dade-025             NONFED
19490   1749796  300090789  SFO-2014GAGAS-FY2014-Dade-025             NONFED
19491   1749796  300090789  SFO-2014GAGAS-FY2014-Dade-025             NONFED

      SOURCE_SYSTEM NWCG_REPORTING_AGENCY NWCG_REPORTING_UNIT_ID  \
0           ST-NASF                 ST/C&L                USWIWIS
1           ST-NASF                 ST/C&L                USWIWIS
2           ST-NASF                 ST/C&L                USWIWIS
3           ST-NASF                 ST/C&L                USWIWIS
4           ST-NASF                 ST/C&L                USWIWIS
…                 …                     …                      …
19487       ST-NASF                 ST/C&L                USGAGAS
19488       ST-NASF                 ST/C&L                USGAGAS
19489       ST-NASF                 ST/C&L                USGAGAS
19490       ST-NASF                 ST/C&L                USGAGAS
19491       ST-NASF                 ST/C&L                USGAGAS

                       NWCG_REPORTING_UNIT_NAME SOURCE_REPORTING_UNIT  \
0       Wisconsin Department of Natural Resources                 WIDNR
```

```
1      Wisconsin Department of Natural Resources                 WIDNR
2      Wisconsin Department of Natural Resources                 WIDNR
3      Wisconsin Department of Natural Resources                 WIDNR
4      Wisconsin Department of Natural Resources                 WIDNR
…                                            …                   …
19487                 Georgia Forestry Commission                 GAGAS
19488                 Georgia Forestry Commission                 GAGAS
19489                 Georgia Forestry Commission                 GAGAS
19490                 Georgia Forestry Commission                 GAGAS
19491                 Georgia Forestry Commission                 GAGAS

                       SOURCE_REPORTING_UNIT_NAME  …   City_Lon  \
0      Wisconsin Department of Natural Resources  …  -93.26384
1      Wisconsin Department of Natural Resources  …  -93.26384
2      Wisconsin Department of Natural Resources  …  -93.26384
3      Wisconsin Department of Natural Resources  …  -93.26384
4      Wisconsin Department of Natural Resources  …  -93.26384
…                                            …  …          …
19487                 Georgia Forestry Commission  … -84.387978
19488                 Georgia Forestry Commission  … -84.387978
19489                 Georgia Forestry Commission  … -84.387978
19490                 Georgia Forestry Commission  … -84.387978
19491                 Georgia Forestry Commission  … -84.387978

      Distance_Miles       discovery_date             Datetime         City  \
0          67.038499  2012-10-04 00:00:00  2012-10-03 12:00:00  Minneapolis
1          67.038499  2012-10-04 00:00:00  2012-10-03 13:00:00  Minneapolis
2          67.038499  2012-10-04 00:00:00  2012-10-03 14:00:00  Minneapolis
3          67.038499  2012-10-04 00:00:00  2012-10-03 15:00:00  Minneapolis
4          67.038499  2012-10-04 00:00:00  2012-10-03 16:00:00  Minneapolis
…                …                  …                  …          …
19487      99.742864  2014-01-30 00:00:00  2014-01-30 08:00:00      Atlanta
19488      99.742864  2014-01-30 00:00:00  2014-01-30 09:00:00      Atlanta
19489      99.742864  2014-01-30 00:00:00  2014-01-30 10:00:00      Atlanta
19490      99.742864  2014-01-30 00:00:00  2014-01-30 11:00:00      Atlanta
19491      99.742864  2014-01-30 00:00:00  2014-01-30 12:00:00      Atlanta

      Humidity Wind_Direction Temperature Pressure Wind_Speed
0         84.0            0.0     278.175   1015.0        0.0
1         75.0            0.0      278.12   1015.0        0.0
2         57.0            0.0      282.48   1015.0        0.0
3         53.0           63.0  285.643333   1014.0        1.0
4         49.0          126.0  288.806667   1014.0        2.0
…          …             …          …        …         …
19487     85.0            0.0      260.65   1026.0        0.0
19488     84.0            0.0      260.29   1026.0        0.0
19489     84.0            0.0      259.94   1026.0        0.0
```

```
19490      77.0            0.0       259.86    1026.0        0.0
19491      77.0            0.0        259.5    1027.0        0.0

[19492 rows x 53 columns]
```

0.3.3 FIRE_SIZE_CLASS_encoded is transferred from FIRE_SIZE_CLASS, in which A is least severe fire, G is most severe fire

0.3.4 Note that the only fire sizes in the final dataset are A, B, C because of merging dataframes

[30]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
final_df['FIRE_SIZE_CLASS_encoded'] = label_encoder.
 ↪fit_transform(final_df['FIRE_SIZE_CLASS'])
```

[31]:
```
columns_to_convert = ["Humidity", "Wind_Direction", "Temperature", "Pressure",␣
 ↪"Wind_Speed"]
final_df[columns_to_convert] = final_df[columns_to_convert].astype(float)
final_df.to_csv('fire_df.csv')
```

[32]:
```
final_df.columns
```

[32]:
```
Index(['OBJECTID', 'FOD_ID', 'FPA_ID', 'SOURCE_SYSTEM_TYPE', 'SOURCE_SYSTEM',
       'NWCG_REPORTING_AGENCY', 'NWCG_REPORTING_UNIT_ID',
       'NWCG_REPORTING_UNIT_NAME', 'SOURCE_REPORTING_UNIT',
       'SOURCE_REPORTING_UNIT_NAME', 'LOCAL_FIRE_REPORT_ID',
       'LOCAL_INCIDENT_ID', 'FIRE_CODE', 'FIRE_NAME',
       'ICS_209_INCIDENT_NUMBER', 'ICS_209_NAME', 'MTBS_ID', 'MTBS_FIRE_NAME',
       'COMPLEX_NAME', 'FIRE_YEAR', 'DISCOVERY_DATE', 'DISCOVERY_DOY',
       'DISCOVERY_TIME', 'STAT_CAUSE_CODE', 'STAT_CAUSE_DESCR', 'CONT_DATE',
       'CONT_DOY', 'CONT_TIME', 'FIRE_SIZE', 'FIRE_SIZE_CLASS', 'LATITUDE',
       'LONGITUDE', 'OWNER_CODE', 'OWNER_DESCR', 'STATE', 'COUNTY',
       'FIPS_CODE', 'FIPS_NAME', 'Shape', 'Fire_Lat', 'Fire_Lon', 'City',
       'City_Lat', 'City_Lon', 'Distance_Miles', 'discovery_date', 'Datetime',
       'City', 'Humidity', 'Wind_Direction', 'Temperature', 'Pressure',
       'Wind_Speed', 'FIRE_SIZE_CLASS_encoded'],
      dtype='object')
```

[33]:
```
type(final_df['Wind_Speed'][0])
```

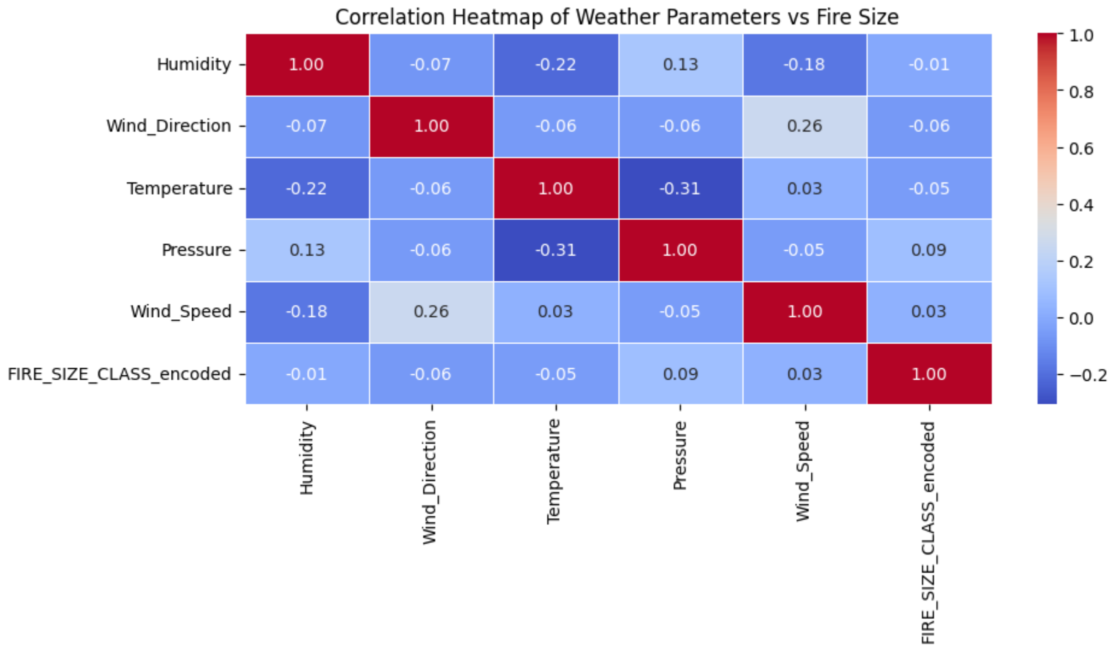
[33]:
```
numpy.float64
```

0.3.5 finally we make a correlation matrix

[34]:
```
numeric_columns = final_df.select_dtypes(include=['number'])

# Compute the correlation matrix
correlation_matrix = numeric_columns.corr()

# Display the correlation matrix
print(correlation_matrix)
```

```
                       Humidity  Wind_Direction  Temperature  Pressure  \
Humidity               1.000000       -0.074717    -0.218958  0.125273
Wind_Direction        -0.074717        1.000000    -0.061926 -0.060450
Temperature           -0.218958       -0.061926     1.000000 -0.308319
Pressure               0.125273       -0.060450    -0.308319  1.000000
Wind_Speed            -0.176721        0.255952     0.028667 -0.054814
FIRE_SIZE_CLASS_encoded -0.009128       -0.064364    -0.053552  0.089604

                       Wind_Speed  FIRE_SIZE_CLASS_encoded
Humidity                -0.176721                -0.009128
Wind_Direction           0.255952                -0.064364
Temperature              0.028667                -0.053552
Pressure                -0.054814                 0.089604
Wind_Speed               1.000000                 0.027888
FIRE_SIZE_CLASS_encoded  0.027888                 1.000000
```

[35]:
```
plt.figure(figsize=(10, 4))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f",␣
↪linewidths=0.5)
plt.title("Correlation Heatmap of Weather Parameters vs Fire Size")
plt.show()
```



0.3.6 Creating Regression Models

Regressing fire size (ordinal) on weather and location features (continuous) using ordinal logistic regression Because fire size is an ordinal categorical variable with 3 different levels, we can run an ordinal logistic regression model (Proportional Odds Model).

[36]:
```
import statsmodels
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import StandardScaler
from statsmodels.miscmodels.ordinal_model import OrderedModel
import seaborn as sns
import matplotlib.pyplot as plt
```

[37]:
```
X = final_df[['Humidity', 'Wind_Direction', 'Temperature', 'Pressure',␣
↪'Wind_Speed', 'City_Lat', 'City_Lon', 'FIRE_SIZE_CLASS_encoded']]
features = X.copy()

# Joining Features with City Latitude and Longitude
final_df[["City_Lat", "City_Lon"]] = final_df[["City_Lat", "City_Lon"]].
↪astype(float)
features = features.drop(columns=['City_Lat', 'City_Lon']).
↪join(final_df[['City_Lat', 'City_Lon']])
```

```python
# Checking for variable collinearity
vif_data = pd.DataFrame()
vif_data["Variable"] = features.columns
vif_data["VIF"] = [variance_inflation_factor(features.values, i) for i in␣
 ↪range(features.shape[1])]
print(vif_data)

features
```

```
                 Variable         VIF
0                Humidity    9.470812
1          Wind_Direction    4.568473
2             Temperature  793.037322
3                Pressure  931.746311
4              Wind_Speed    3.069062
5  FIRE_SIZE_CLASS_encoded    1.884162
6                City_Lat   71.160374
7                City_Lon   37.277037
```

[37]:

```
       Humidity  Wind_Direction  Temperature  Pressure  Wind_Speed  \
0          84.0             0.0   278.175000    1015.0         0.0
1          75.0             0.0   278.120000    1015.0         0.0
2          57.0             0.0   282.480000    1015.0         0.0
3          53.0            63.0   285.643333    1014.0         1.0
4          49.0           126.0   288.806667    1014.0         2.0
…           …             …          …          …           …
19487      85.0             0.0   260.650000    1026.0         0.0
19488      84.0             0.0   260.290000    1026.0         0.0
19489      84.0             0.0   259.940000    1026.0         0.0
19490      77.0             0.0   259.860000    1026.0         0.0
19491      77.0             0.0   259.500000    1027.0         0.0

       FIRE_SIZE_CLASS_encoded   City_Lat   City_Lon
0                            1  44.979969 -93.263840
1                            1  44.979969 -93.263840
2                            1  44.979969 -93.263840
3                            1  44.979969 -93.263840
4                            1  44.979969 -93.263840
…                          …         …         …
19487                        0  33.749001 -84.387978
19488                        0  33.749001 -84.387978
19489                        0  33.749001 -84.387978
19490                        0  33.749001 -84.387978
19491                        0  33.749001 -84.387978

[19492 rows x 8 columns]
```

Standardizing Independent Variables that have higher VIFs Keeping latitude and longitude the same because it only has meaning when kept in original scale

[38]:
```
# Define features to standardize and keep unstandardized
standardized_features = ['Humidity', 'Temperature', 'Pressure']
nonstandardized_features = ['Wind_Direction', 'Wind_Speed', 'City_Lat',␣
↪'City_Lon']

# Standardize selected features
scaler = StandardScaler()
standardized_array = scaler.fit_transform(final_df[standardized_features])

# Convert scaled features into DataFrame with correct index
standardized_df = pd.DataFrame(standardized_array,␣
↪columns=[f"{col}_standardized" for col in standardized_features],␣
↪index=final_df.index)

# Join with unstandardized features
features_scaled = standardized_df.join(final_df[nonstandardized_features])

# Check the final DataFrame
features_scaled.head()
```

[38]:
```
   Humidity_standardized  Temperature_standardized  Pressure_standardized  \
0               1.100104                 -1.119274              -0.486880
1               0.684975                 -1.124882              -0.486880
2              -0.145281                 -0.680272              -0.486880
3              -0.329783                 -0.357692              -0.576293
4              -0.514284                 -0.035112              -0.576293

   Wind_Direction  Wind_Speed   City_Lat  City_Lon
0             0.0         0.0  44.979969 -93.26384
1             0.0         0.0  44.979969 -93.26384
2             0.0         0.0  44.979969 -93.26384
3            63.0         1.0  44.979969 -93.26384
4           126.0         2.0  44.979969 -93.26384
```

[39]:
```
# Checking for variable collinearity after scaling
vif_data = pd.DataFrame()
vif_data["Variable"] = features_scaled.columns
vif_data["VIF"] = [variance_inflation_factor(features_scaled.values, i) for i␣
↪in range(features_scaled.shape[1])]
print(vif_data)
```

```
                  Variable       VIF
0    Humidity_standardized  1.087367
1 Temperature_standardized  1.254863
2    Pressure_standardized  1.122999
```

```
3           Wind_Direction   4.591735
4               Wind_Speed   2.942939
5                 City_Lat  26.956378
6                 City_Lon  23.273701
```

Dropping City_Lon from X variables because City Latitude is likely more important to prediction than City Longitude since Latitude is on average associated average location temperature (cities closer to equator have higher temperatures).

[40]:
```python
# Dropping City_Lon from X variables
features_scaled = features_scaled.drop(columns=['City_Lon'])
print("predictor variables: ")

vif_data = pd.DataFrame()
vif_data["Variable"] = features_scaled.columns
vif_data["VIF"] = [variance_inflation_factor(features_scaled.values, i) for i␣
 ↪in range(features_scaled.shape[1])]
print(vif_data)
```

```
predictor variables:
                   Variable       VIF
0     Humidity_standardized  1.087349
1  Temperature_standardized  1.151861
2     Pressure_standardized  1.116165
3            Wind_Direction  4.562096
4                Wind_Speed  2.920621
5                  City_Lat  5.033826
```

No issues with multicollinearity now!

[41]:
```python
multinomial_df = features_scaled.join(final_df["FIRE_SIZE_CLASS_encoded"])
multinomial_df
```

[41]:
```
       Humidity_standardized  Temperature_standardized  Pressure_standardized  \
0                   1.100104                 -1.119274              -0.486880
1                   0.684975                 -1.124882              -0.486880
2                  -0.145281                 -0.680272              -0.486880
3                  -0.329783                 -0.357692              -0.576293
4                  -0.514284                 -0.035112              -0.576293
…                        …                         …                      …
19487               1.146229                 -2.906382               0.496660
19488               1.100104                 -2.943092               0.496660
19489               1.100104                 -2.978784               0.496660
19490               0.777226                 -2.986942               0.496660
19491               0.777226                 -3.023653               0.586073

       Wind_Direction  Wind_Speed   City_Lat  FIRE_SIZE_CLASS_encoded
0                 0.0         0.0  44.979969                        1
1                 0.0         0.0  44.979969                        1
```

```
2                 0.0          0.0  44.979969                        1
3                63.0          1.0  44.979969                        1
4               126.0          2.0  44.979969                        1
…                  …          …        …                    …
19487             0.0          0.0  33.749001                        0
19488             0.0          0.0  33.749001                        0
19489             0.0          0.0  33.749001                        0
19490             0.0          0.0  33.749001                        0
19491             0.0          0.0  33.749001                        0

[19492 rows x 7 columns]
```

[43]:
```python
multinomial_df["FIRE_SIZE_CLASS_encoded"].unique()
```

[43]:
```
array([1, 0, 2, 3, 4])
```

[42]:
```python
# Ordered Logistic Regression
formula = 'FIRE_SIZE_CLASS_encoded ~ Humidity_standardized +␣
 ↪Temperature_standardized + Pressure_standardized + Wind_Direction +␣
 ↪Wind_Speed + City_Lat'
model = OrderedModel.from_formula(
    formula,
    data=multinomial_df,
    distr='logit'
)
result = model.fit(method='bfgs', maxiter=1000) #bfgs method uses Gradients to␣
 ↪compute Hessian instead of directly computing, which allows for faster␣
 ↪convergence

print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.893174
         Iterations: 30
         Function evaluations: 34
         Gradient evaluations: 34
                             OrderedModel Results
================================================================================
===
Dep. Variable:     FIRE_SIZE_CLASS_encoded   Log-Likelihood:
-17410.
Model:                        OrderedModel   AIC:
3.484e+04
Method:                 Maximum Likelihood   BIC:
3.492e+04
Date:                     Sat, 15 Mar 2025
Time:                             23:33:34
No. Observations:                    19492
```

```
Df Residuals:                         19482
Df Model:                                 6
================================================================================
============
                              coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
------------
Humidity_standardized      -0.0592      0.015     -3.960      0.000      -0.089
-0.030
Temperature_standardized   -0.2472      0.016    -15.559      0.000      -0.278
-0.216
Pressure_standardized       0.1881      0.017     10.944      0.000       0.154
0.222
Wind_Direction             -0.0007      0.000     -4.709      0.000      -0.001
-0.000
Wind_Speed                  0.0590      0.007      7.907      0.000       0.044
0.074
City_Lat                   -0.1216      0.003    -35.244      0.000      -0.128
-0.115
0.0/1.0                    -4.3367      0.126    -34.489      0.000      -4.583
-4.090
1.0/2.0                     0.9553      0.011     88.595      0.000       0.934
0.976
2.0/3.0                     0.7283      0.032     22.655      0.000       0.665
0.791
3.0/4.0                    -0.0127      0.093     -0.136      0.892      -0.195
0.169
================================================================================
============
```

Coefficient Interpretations: Variables that significantly affect fire size - Higher Pressure associated with Larger Fire Size: a one standard deviation increase in pressure increases the log odds of a larger fire size category by 0.6132. ($p < 0.001$, strongest effect). - Higher Humidity associated Larger Fire Size: a one standard deviation increase in humidity increases the log odds of being in a larger fire size category by 0.1083 ($p = 0.053$, weak effect). **Does not make sense** - Higher Wind Speed associated with Smaller Fire Size: a one standard deviation decrease in wind speed increases the log odds of being in a larger fire size category by 0.1083 ($p = 0.007$, moderate effect). **Also does not make sense** - Lower Latitude: A one degree increase in latitude decreases the fire size log-odds by 0.2030

Variables that do not have a strong relationship with fire size at significance level of 0.05 - Temperature - Wind Direction

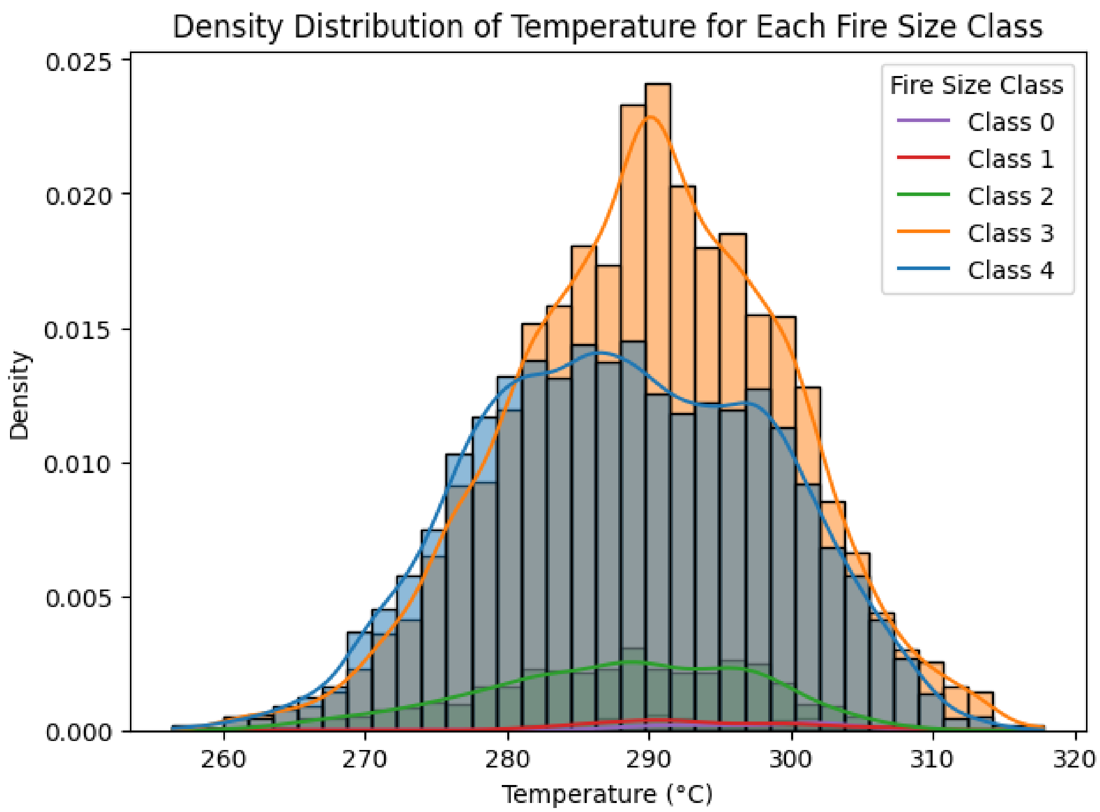
Lots of coefficients that don't make sense

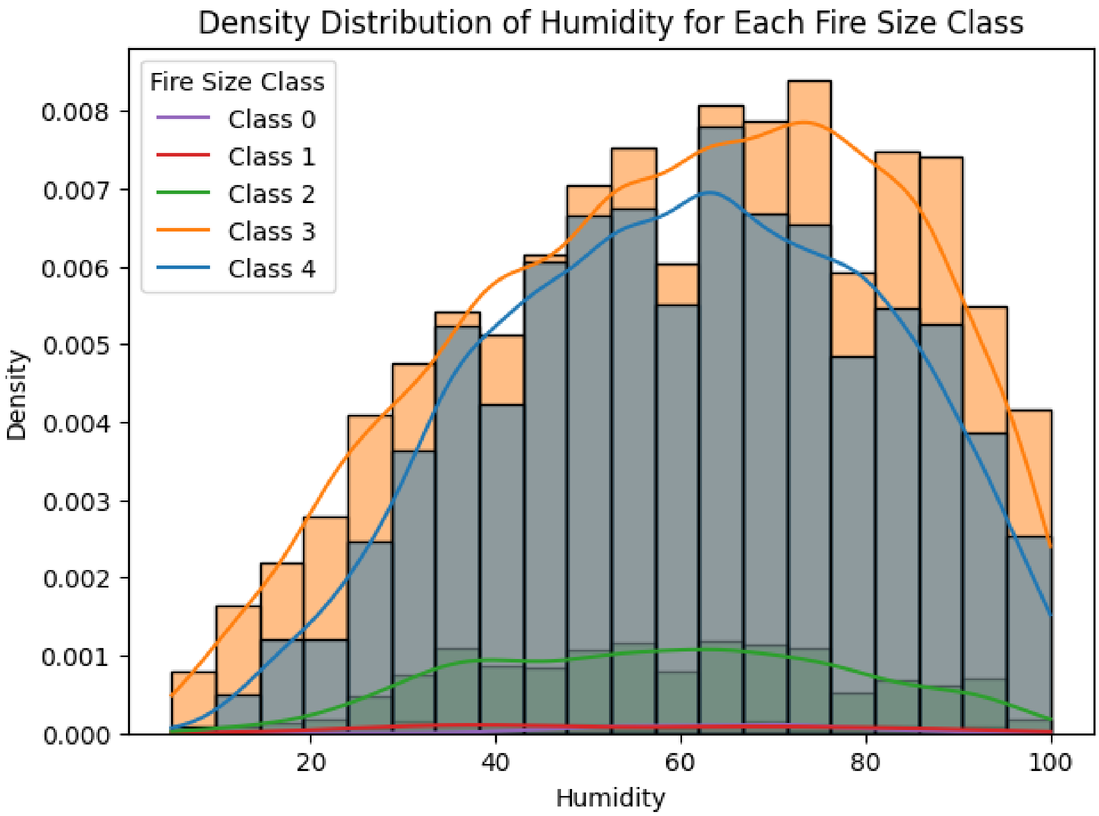
0.3.7 Plotting the Data below to see why the coefficients of humidity and temperature are counterintuitive

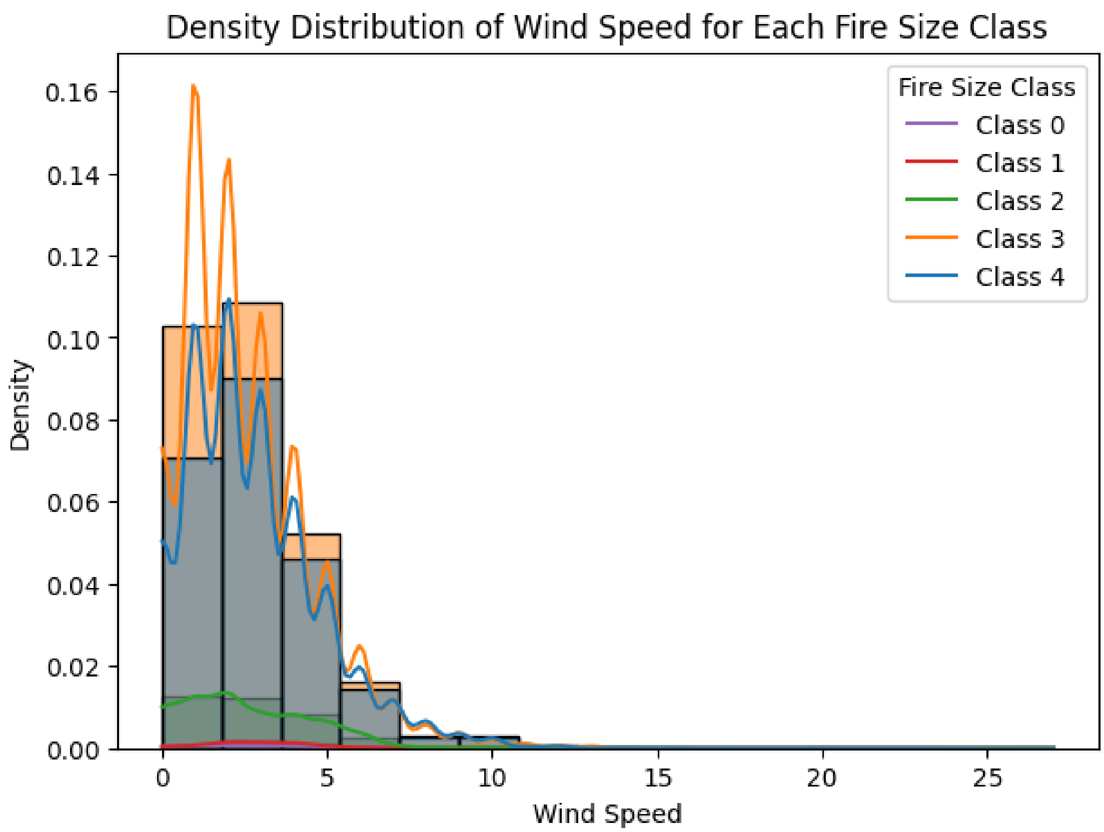
[46]:

```
# density plot for temperature distribution by fire size class
plt.figure(figsize=(7, 5))
sns.histplot(
    data=final_df,
    x="Temperature",
    hue=final_df["FIRE_SIZE_CLASS_encoded"].astype(str),
    kde=True,
    bins=35,
    stat="density"
)

plt.xlabel("Temperature (°C)")
plt.ylabel("Density")
plt.title("Density Distribution of Temperature for Each Fire Size Class")
plt.legend(title="Fire Size Class", labels=["Class 0", "Class 1", "Class 2",␣
 ↪"Class 3", "Class 4"])
plt.show()

# density plot for humidity distribution by fire size class
plt.figure(figsize=(7, 5))
sns.histplot(
    data=final_df,
    x="Humidity",
    hue=final_df["FIRE_SIZE_CLASS_encoded"].astype(str),
    kde=True,
    bins=20,
    stat="density"
)

plt.xlabel("Humidity")
plt.ylabel("Density")
plt.title("Density Distribution of Humidity for Each Fire Size Class")
plt.legend(title="Fire Size Class", labels=["Class 0", "Class 1", "Class 2",␣
 ↪"Class 3", "Class 4"])
plt.show()

# density plot for wind speed distribution by fire size class
plt.figure(figsize=(7, 5))
sns.histplot(
    data=final_df,
    x="Wind_Speed",
    hue=final_df["FIRE_SIZE_CLASS_encoded"].astype(str),
    kde=True,
    bins=15,
```

```
    stat="density"
)

plt.xlabel("Wind Speed")
plt.ylabel("Density")
plt.title("Density Distribution of Wind Speed for Each Fire Size Class")
plt.legend(title="Fire Size Class", labels=["Class 0", "Class 1", "Class 2",␣
↪"Class 3", "Class 4"])
plt.show()
```




Density Distribution of Humidity for Each Fire Size Class
Fire Size Class
Class 0
Class 1
Class 2
Class 3
Class 4
Density
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
20
40
60
80
100
Humidity



[48]:
```
temp_humidity_df = pd.DataFrame()

for fire_class in [0,1,2,3,4]:
    queried_df = final_df[final_df["FIRE_SIZE_CLASS_encoded"] == fire_class]
    avg_class_humidity = sum(queried_df["Humidity"])/len(queried_df["Humidity"])
    avg_class_temp = sum(queried_df["Temperature"])/
 ↪len(queried_df["Temperature"])
    avg_class_speed = sum(queried_df["Wind_Speed"])/
 ↪len(queried_df["Wind_Speed"])

    temp_humidity_df.loc[fire_class, "Average Temperature"] = avg_class_temp
    temp_humidity_df.loc[fire_class, "Average Humidity"] = avg_class_humidity
    temp_humidity_df.loc[fire_class, "Average Wind Speed"] = avg_class_speed

temp_humidity_df.index.name = "Fire Size Class"

temp_humidity_df
```

[48]:
```
                    Average Temperature  Average Humidity  Average Wind Speed
Fire Size Class
```

```
0                          289.958203          60.112850              2.525371
1                          288.212169          60.558706              2.717712
2                          287.765363          58.117928              2.515538
3                          293.666294          55.072000              2.880000
4                          297.064084          63.880000              2.493333
```

Seems like average humidity increases as class size increases in the data, which doesn't make sense scientifically but matches our model.

0.3.8 Trying interaction terms:

shows that the effect of temperature on fire size becomes stronger (more positive) as humidity increases which matches what we see happening in the data above.

[49]:
```
formula = 'FIRE_SIZE_CLASS_encoded ~ Temperature_standardized *␣
↪Humidity_standardized + Pressure_standardized + Wind_Direction + Wind_Speed␣
↪+ City_Lat'
model = OrderedModel.from_formula(
    formula,
    data=multinomial_df,
    distr='logit'
)
result = model.fit(method='bfgs', maxiter=1000) #bfgs method uses Gradients to␣
↪compute Hessian instead of directly computing, which allows for faster␣
↪convergence

print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.890394
         Iterations: 40
         Function evaluations: 44
         Gradient evaluations: 44
                                OrderedModel Results
================================================================================
===
Dep. Variable:     FIRE_SIZE_CLASS_encoded   Log-Likelihood:
-17356.
Model:                        OrderedModel   AIC:
3.473e+04
Method:                 Maximum Likelihood   BIC:
3.482e+04
Date:                     Sun, 16 Mar 2025
Time:                             00:39:20
No. Observations:                    19492
Df Residuals:                        19481
Df Model:                                7
```

```
================================================================================
==================================
                                                           coef    std err          z
P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
----------------------------------
Temperature_standardized                                -0.2151      0.016    -13.275
0.000      -0.247      -0.183
Humidity_standardized                                   -0.0713      0.015     -4.734
0.000      -0.101      -0.042
Temperature_standardized:Humidity_standardized           0.1547      0.015     10.308
0.000       0.125       0.184
Pressure_standardized                                    0.1915      0.017     11.105
0.000       0.158       0.225
Wind_Direction                                          -0.0007      0.000     -5.152
0.000      -0.001      -0.000
Wind_Speed                                               0.0523      0.007      6.983
0.000       0.038       0.067
City_Lat                                                -0.1204      0.003    -34.929
0.000      -0.127      -0.114
0.0/1.0                                                 -4.3576      0.126    -34.660
0.000      -4.604      -4.111
1.0/2.0                                                  0.9591      0.011     89.000
0.000       0.938       0.980
2.0/3.0                                                  0.7296      0.032     22.717
0.000       0.667       0.793
3.0/4.0                                                 -0.0109      0.093     -0.118
0.906      -0.193       0.171
================================================================================
==================================
```

0.3.9 next we use other prediction methods (Yiming 0315)

[83]:
```
prediction_df = final_df[["Fire_Lat","Fire_Lon","Datetime",'Humidity',␣
↪'Wind_Direction', 'Temperature', 'Pressure',
       'Wind_Speed', 'FIRE_SIZE_CLASS_encoded']]
prediction_df
```

[83]:
```
        Fire_Lat   Fire_Lon            Datetime  Humidity  Wind_Direction  \
0       45.70474  -92.34605 2012-10-03 12:00:00      84.0             0.0
1       45.70474  -92.34605 2012-10-03 13:00:00      75.0             0.0
2       45.70474  -92.34605 2012-10-03 14:00:00      57.0             0.0
3       45.70474  -92.34605 2012-10-03 15:00:00      53.0            63.0
4       45.70474  -92.34605 2012-10-03 16:00:00      49.0           126.0
…            …          …                   …         …               …
19487  34.846072 -85.523808 2014-01-30 08:00:00      85.0             0.0
```

```
19488  34.846072 -85.523808  2014-01-30 09:00:00      84.0             0.0
19489  34.846072 -85.523808  2014-01-30 10:00:00      84.0             0.0
19490  34.846072 -85.523808  2014-01-30 11:00:00      77.0             0.0
19491  34.846072 -85.523808  2014-01-30 12:00:00      77.0             0.0

       Temperature  Pressure  Wind_Speed  FIRE_SIZE_CLASS_encoded
0       278.175000    1015.0         0.0                        1
1       278.120000    1015.0         0.0                        1
2       282.480000    1015.0         0.0                        1
3       285.643333    1014.0         1.0                        1
4       288.806667    1014.0         2.0                        1
…              …         …          …                      …
19487   260.650000    1026.0         0.0                        0
19488   260.290000    1026.0         0.0                        0
19489   259.940000    1026.0         0.0                        0
19490   259.860000    1026.0         0.0                        0
19491   259.500000    1027.0         0.0                        0

[19492 rows x 9 columns]
```

[101]:

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler,␣
↪FunctionTransformer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

df = prediction_df.copy()
df["Datetime"] = pd.to_datetime(df["Datetime"])
df["hour"] = df["Datetime"].dt.hour
df["month"] = df["Datetime"].dt.month
df["is_daytime"] = df["hour"].between(6, 18).astype(int)

df = df.drop(columns=["Datetime"])

X = df.drop(columns="FIRE_SIZE_CLASS_encoded")
y = df["FIRE_SIZE_CLASS_encoded"]

numeric_features = ["Humidity", "Temperature", "Pressure", "Wind_Speed"]
cyclical_features = ["hour", "Wind_Direction"]

# Cyclical encoding for hour and wind direction
def cyclical_encoding(X, col, max_val):
    return np.sin(2 * np.pi * X[col].to_numpy() / max_val).reshape(-1, 1)  #␣
↪Convert to NumPy array and reshape
```

```
preprocessor = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), numeric_features),
        ("cyclical_hour", FunctionTransformer(lambda x: cyclical_encoding(x,␣
 ↪"hour", 24)), ["hour"]),
        ("cyclical_wind", FunctionTransformer(lambda x: cyclical_encoding(x,␣
 ↪"Wind_Direction", 360)), ["Wind_Direction"])
    ]
)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪stratify=y)
```

[102]:
```
model = Pipeline([
    ("preprocessor", preprocessor),
    ("classifier", RandomForestClassifier(class_weight="balanced"))
])
model.fit(X_train, y_train)
```

[102]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num', StandardScaler(),
                                                  ['Humidity', 'Temperature',
                                                   'Pressure', 'Wind_Speed']),
                                                 ('cyclical_hour',
FunctionTransformer(func=<function <lambda> at 0x3abc0b1a0>),
                                                  ['hour']),
                                                 ('cyclical_wind',
FunctionTransformer(func=<function <lambda> at 0x3dc932f20>),
                                                  ['Wind_Direction'])])),
                ('classifier',
                 RandomForestClassifier(class_weight='balanced'))])
```

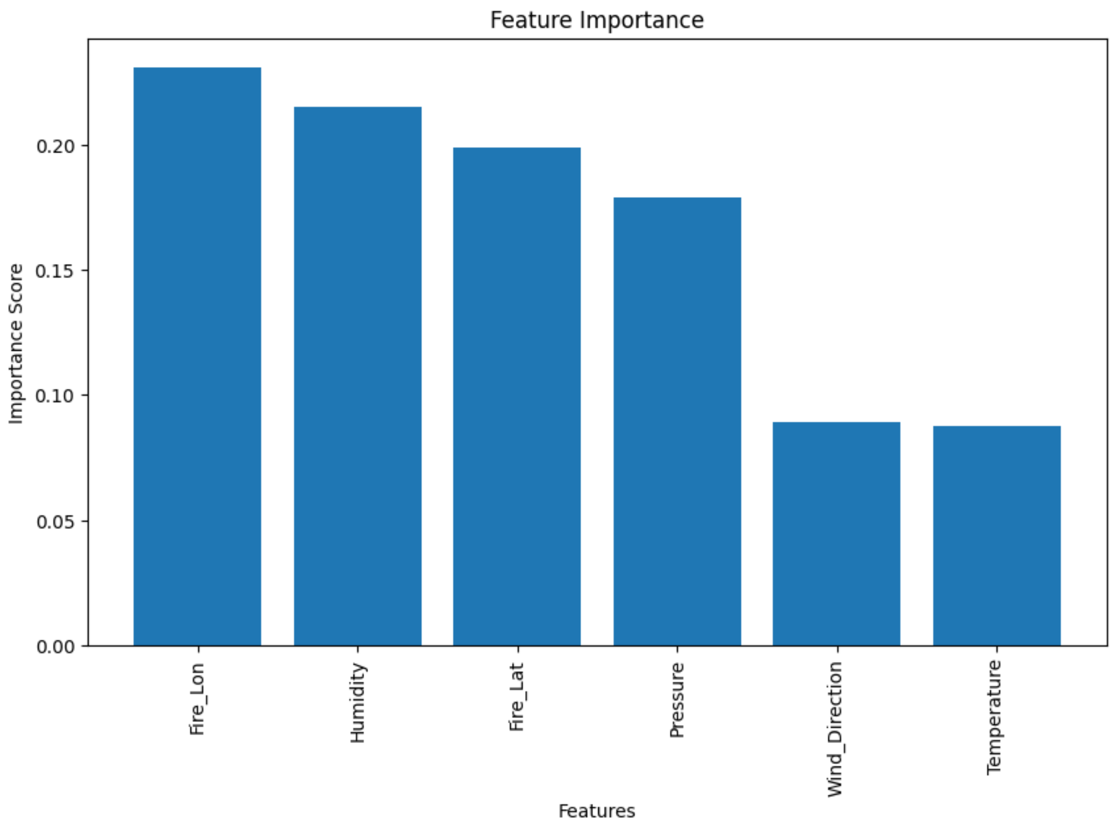
using standard 80%-20% train test split, we achieve 88% prediction accuracy using random forest model

[103]:
```
print("Accuracy:", model.score(X_test, y_test))
```

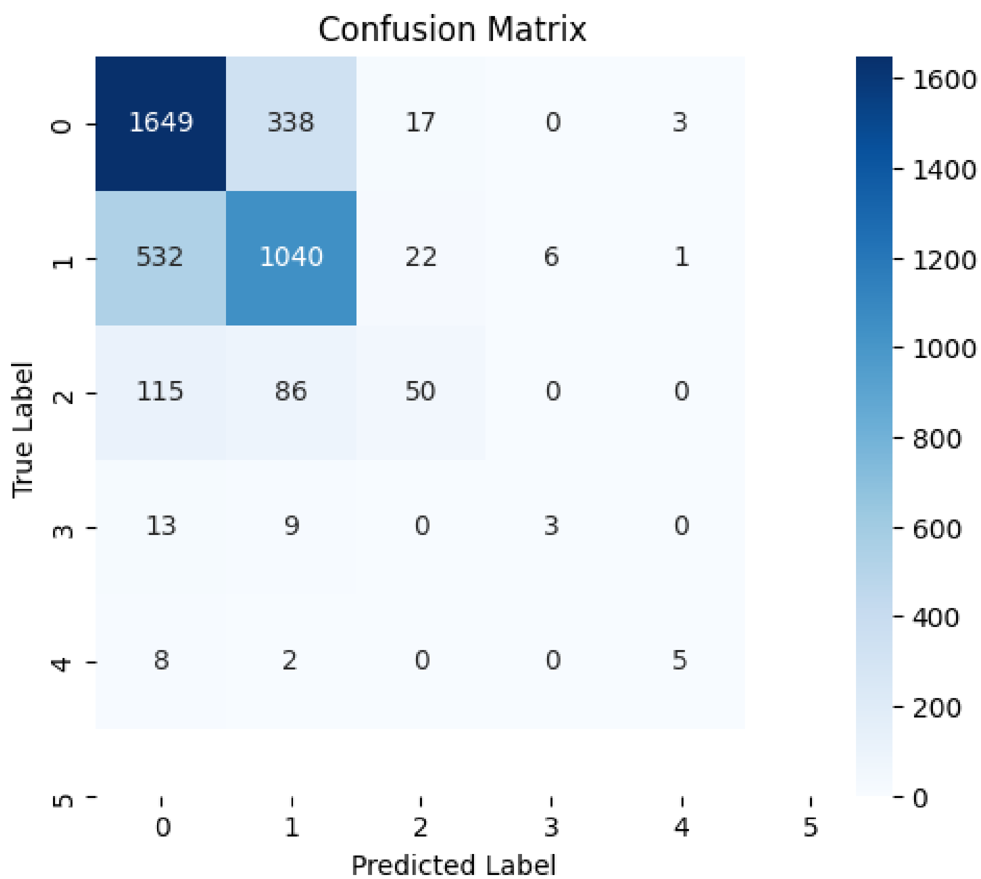
```
Accuracy: 0.7045396255450116
```

[104]:
```
importances = model.named_steps["classifier"].feature_importances_
feature_names = X_train.columns

indices = np.argsort(importances)[::-1]
plt.figure(figsize=(10, 6))
plt.title("Feature Importance")
plt.bar(range(len(importances)), importances[indices], align="center")
plt.xticks(range(len(importances)), feature_names[indices], rotation=90)
plt.xlabel("Features")
```

```
plt.ylabel("Importance Score")
plt.show()
```



[105]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns

y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["0", "1","2",␣
↪"3", "4", "5"], yticklabels=["0", "1","2", "3", "4", "5"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```



0.3.10 Building a Neural Network Model

[91]:
```
pip install torch
```

```
Collecting torch
  Downloading torch-2.6.0-cp313-none-macosx_11_0_arm64.whl.metadata (28 kB)
Collecting filelock (from torch)
  Downloading filelock-3.18.0-py3-none-any.whl.metadata (2.9 kB)
Requirement already satisfied: typing-extensions>=4.10.0 in
/opt/homebrew/Cellar/jupyterlab/4.3.4_1/libexec/lib/python3.13/site-packages
(from torch) (4.12.2)
Collecting networkx (from torch)
  Downloading networkx-3.4.2-py3-none-any.whl.metadata (6.3 kB)
Requirement already satisfied: jinja2 in
/opt/homebrew/Cellar/jupyterlab/4.3.4_1/libexec/lib/python3.13/site-packages
(from torch) (3.1.5)
Collecting fsspec (from torch)
  Downloading fsspec-2025.3.0-py3-none-any.whl.metadata (11 kB)
```

```
Requirement already satisfied: setuptools in
/opt/homebrew/Cellar/jupyterlab/4.3.4_1/libexec/lib/python3.13/site-packages
(from torch) (75.6.0)
Collecting sympy==1.13.1 (from torch)
  Downloading sympy-1.13.1-py3-none-any.whl.metadata (12 kB)
Collecting mpmath<1.4,>=1.1.0 (from sympy==1.13.1->torch)
  Downloading mpmath-1.3.0-py3-none-any.whl.metadata (8.6 kB)
Requirement already satisfied: MarkupSafe>=2.0 in
/opt/homebrew/Cellar/jupyterlab/4.3.4_1/libexec/lib/python3.13/site-packages
(from jinja2->torch) (3.0.2)
Downloading torch-2.6.0-cp313-none-macosx_11_0_arm64.whl (66.5 MB)
                         66.5/66.5 MB
25.4 MB/s eta 0:00:00a 0:00:01
Downloading sympy-1.13.1-py3-none-any.whl (6.2 MB)
                         6.2/6.2 MB
23.0 MB/s eta 0:00:00a 0:00:01
Downloading filelock-3.18.0-py3-none-any.whl (16 kB)
Downloading fsspec-2025.3.0-py3-none-any.whl (193 kB)
Downloading networkx-3.4.2-py3-none-any.whl (1.7 MB)
                         1.7/1.7 MB
27.4 MB/s eta 0:00:00
Downloading mpmath-1.3.0-py3-none-any.whl (536 kB)
                         536.2/536.2 kB
16.0 MB/s eta 0:00:00
Installing collected packages: mpmath, sympy, networkx, fsspec, filelock, torch
Successfully installed filelock-3.18.0 fsspec-2025.3.0 mpmath-1.3.0
networkx-3.4.2 sympy-1.13.1 torch-2.6.0

[notice] A new release of pip is
available: 24.3.1 -> 25.0.1
[notice] To update, run:
/opt/homebrew/Cellar/jupyterlab/4.3.4_1/libexec/bin/python -m pip

install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

[106]:

```
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, FunctionTransformer
from sklearn.compose import ColumnTransformer
from sklearn.utils.class_weight import compute_class_weight
import matplotlib.pyplot as plt
```

```
df = prediction_df.copy()
df["Datetime"] = pd.to_datetime(df["Datetime"])
df["hour"] = df["Datetime"].dt.hour
df["month"] = df["Datetime"].dt.month
df["is_daytime"] = df["hour"].between(6, 18).astype(int)

df = df.drop(columns=["Datetime"])
X = df.drop(columns="FIRE_SIZE_CLASS_encoded")
y = df["FIRE_SIZE_CLASS_encoded"].values
numeric_features = ["Humidity", "Temperature", "Pressure", "Wind_Speed"]

def cyclical_encoding(X, col, max_val):
    return np.sin(2 * np.pi * X[col].to_numpy() / max_val).reshape(-1, 1)

preprocessor = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), numeric_features),
        ("cyclical_hour", FunctionTransformer(lambda x: cyclical_encoding(x,␣
 ↪"hour", 24)), ["hour"]),
        ("cyclical_wind", FunctionTransformer(lambda x: cyclical_encoding(x,␣
 ↪"Wind_Direction", 360)), ["Wind_Direction"])
    ]
)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪stratify=y, random_state=42)
```

[107]:
```
X_train_transformed = preprocessor.fit_transform(X_train)
X_test_transformed = preprocessor.transform(X_test)
X_train_tensor = torch.tensor(X_train_transformed, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test_transformed, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.long)
y_test_tensor = torch.tensor(y_test, dtype=torch.long)
train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
test_dataset = TensorDataset(X_test_tensor, y_test_tensor)
train_loader = DataLoader(train_dataset, batch_size=256, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=256, shuffle=False)
```

[108]:
```
class FireSizeClassifier(nn.Module):
    def __init__(self, input_dim, n_classes):
        super(FireSizeClassifier, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(input_dim, 512),
            nn.ReLU(),
            nn.BatchNorm1d(512),
            nn.Dropout(0.6),
```

```
            nn.Linear(512, 256),
            nn.ReLU(),
            nn.BatchNorm1d(256),
            nn.Dropout(0.5),

            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Dropout(0.4),

            nn.Linear(128, 64),
            nn.ReLU(),
            nn.BatchNorm1d(64),
            nn.Dropout(0.3),

            nn.Linear(64, 32),
            nn.ReLU(),

            nn.Linear(32, n_classes)
        )

    def forward(self, x):
        return self.model(x)
```

[109]:
```
# Initialize model
input_dim = X_train_transformed.shape[1]
n_classes = len(np.unique(y))
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = FireSizeClassifier(input_dim, n_classes).to(device)

# Class weights
class_weights = compute_class_weight("balanced", classes=np.unique(y_train),␣
 ↪y=y_train)
class_weights = torch.tensor(class_weights, dtype=torch.float32).to(device)

# Loss, optimizer, and scheduler
criterion = nn.CrossEntropyLoss(weight=class_weights)
optimizer = optim.Adam(model.parameters(), lr=0.01)

# Training loop
early_stopping_patience = 15
best_val_loss = float("inf")
patience_counter = 0
train_losses, val_losses = [], []

for epoch in range(200):
    model.train()
    running_loss = 0.0
```

```
    for X_batch, y_batch in train_loader:
        X_batch, y_batch = X_batch.to(device), y_batch.to(device)
        optimizer.zero_grad()
        outputs = model(X_batch)
        loss = criterion(outputs, y_batch)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

    train_loss = running_loss / len(train_loader)
    train_losses.append(train_loss)

    model.eval()
    val_loss = 0.0
    with torch.no_grad():
        for X_batch, y_batch in test_loader:
            X_batch, y_batch = X_batch.to(device), y_batch.to(device)
            outputs = model(X_batch)
            loss = criterion(outputs, y_batch)
            val_loss += loss.item()
    val_loss /= len(test_loader)
    val_losses.append(val_loss)

    if (epoch + 1) % 10 == 0:
        print(f"Epoch {epoch+1}: Train Loss = {train_loss:.4f}, Val Loss =␣
↪{val_loss:.4f}")

    if val_loss < best_val_loss:
        best_val_loss = val_loss
        patience_counter = 0
        torch.save(model.state_dict(), "best_fire_model.pth")
    else:
        patience_counter += 1
        if patience_counter >= early_stopping_patience:
            print("Early stopping triggered")
            break

model.load_state_dict(torch.load("best_fire_model.pth"))
```

```
Epoch 10: Train Loss = 1.2486, Val Loss = 1.1620
Epoch 20: Train Loss = 1.1384, Val Loss = 1.0673
Epoch 30: Train Loss = 1.1189, Val Loss = 1.0380
Epoch 40: Train Loss = 1.0997, Val Loss = 1.0496
Epoch 50: Train Loss = 1.0500, Val Loss = 1.0333
Epoch 60: Train Loss = 1.0178, Val Loss = 1.0309
Epoch 70: Train Loss = 1.0294, Val Loss = 1.0200
Epoch 80: Train Loss = 1.0198, Val Loss = 0.9792
Epoch 90: Train Loss = 0.9862, Val Loss = 1.0025
```

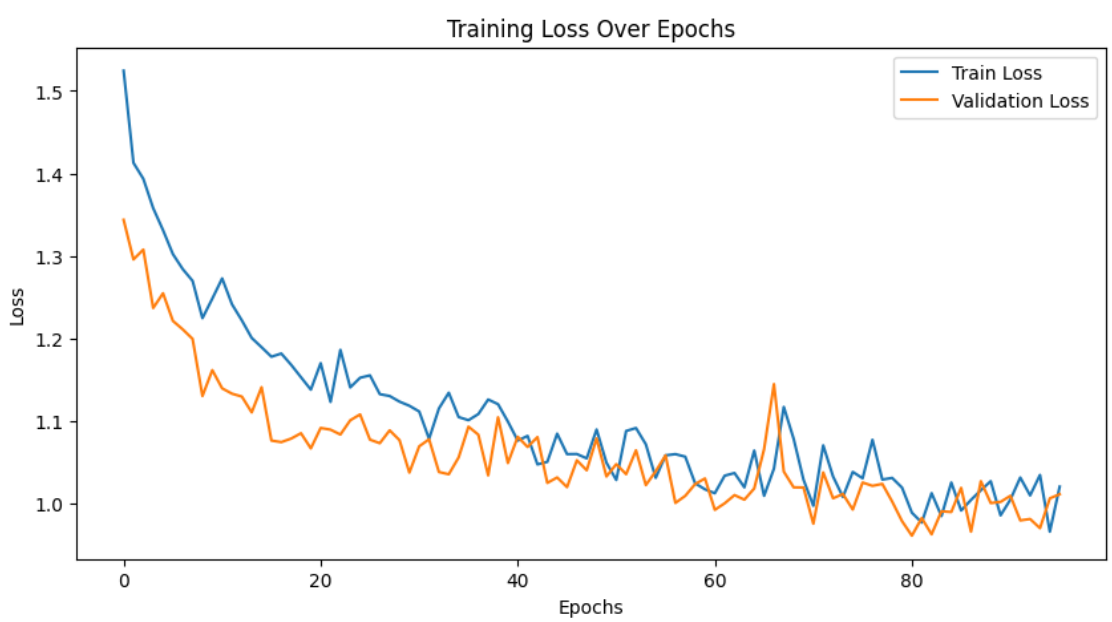
```
Early stopping triggered
```

[109]:
```
&lt;All keys matched successfully&gt;
```

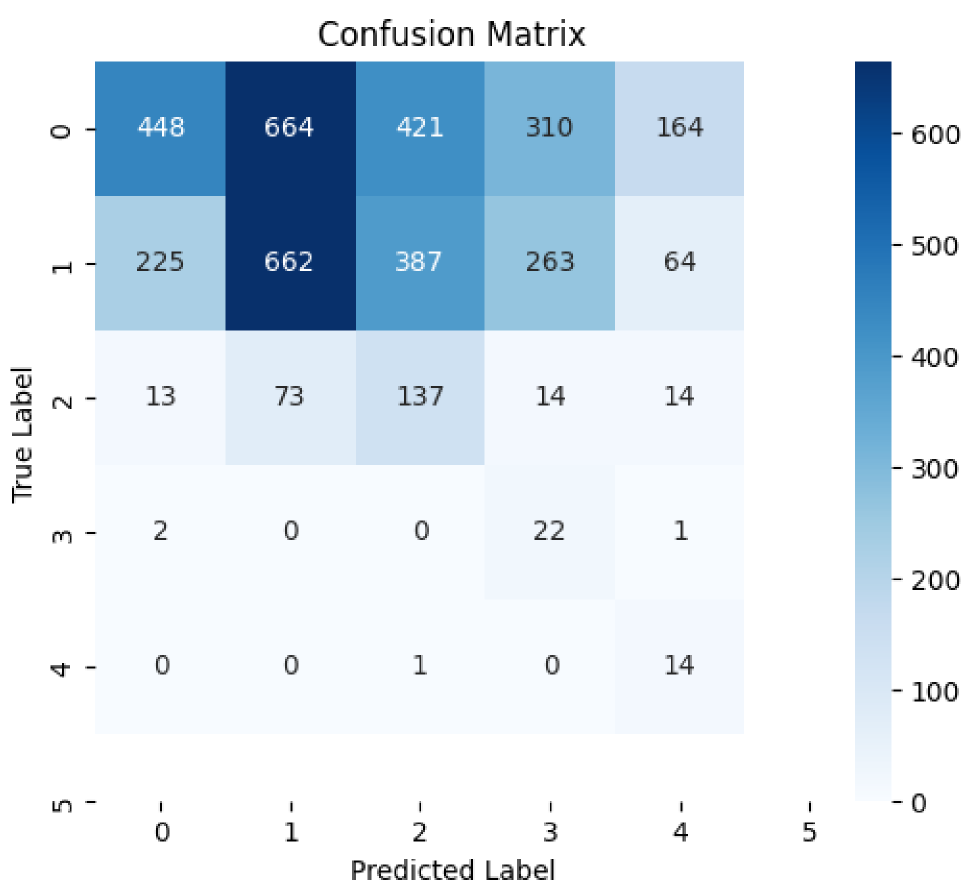
[110]:
```python
model.eval()
correct, total = 0, 0
with torch.no_grad():
    for X_batch, y_batch in test_loader:
        X_batch, y_batch = X_batch.to(device), y_batch.to(device)
        outputs = model(X_batch)
        _, predicted = torch.max(outputs, 1)
        total += y_batch.size(0)
        correct += (predicted == y_batch).sum().item()

test_acc = correct / total
print(f"Test Accuracy: {test_acc:.4f}")
```

```
Test Accuracy: 0.3291
```

[111]:
```python
plt.figure(figsize=(10, 5))
plt.plot(train_losses, label="Train Loss")
plt.plot(val_losses, label="Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.title("Training Loss Over Epochs")
plt.show()
```



[113]:
```
y_true, y_pred = [], []

with torch.no_grad():
    for X_batch, y_batch in test_loader:
        X_batch, y_batch = X_batch.to(device), y_batch.to(device)
        outputs = model(X_batch)
        _, predicted = torch.max(outputs, 1)
        y_true.extend(y_batch.cpu().numpy())
        y_pred.extend(predicted.cpu().numpy())

cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["0", "1","2",␣
 ↪"3", "4", "5"], yticklabels=["0", "1","2", "3", "4", "5"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```



0.3.11 some conclusion: the prediction looks more reasonable in random forest, and we can see the feature importance score is working. However, neural network is not as effective as expected. It might work better with more data

---

0.3.12 Exploratory Data Analysis

[50]:
```
final_df['FIRE_SIZE_CLASS_encoded'].value_counts()
```

[50]:
```
FIRE_SIZE_CLASS_encoded
0    10031
1     8006
2     1255
3      125
4       75
Name: count, dtype: int64
```

[51]:
```
len(final_df['STATE'].value_counts())
```

[51]:
```
32
```

We were curious to see how the sizes of fires varied across the United States. As you can see above, the 'FIRE_SIZE_CLASS_encoded' column has 3 categories that fires are classified into depending on the size of the fires: - 0: (0, 0.25] acres i.e, fires that are greater than 0 but less than or equal to 0.25 acres - 1: [0.26-9.9] acres
- 2: [10.0-99.9] acres

We created a choropleth map that shows the average size of the fires per state across the country. There are 21 U.S. states represented in our dataset, and you can see the average fire sizes in those states below. We chose a continuous color scale to emphasize how Florida, for instance, has larger fires on average than Colorado.

[53]:
```
state_fire_classes = final_df.groupby("STATE",␣
 ↪as_index=False)["FIRE_SIZE_CLASS_encoded"].mean()

fig = px.choropleth(
        state_fire_classes,
        locations="STATE",
        locationmode="USA-states",
        scope="usa",
        color="FIRE_SIZE_CLASS_encoded",
        color_continuous_scale="Reds",
        labels={"FIRE_SIZE_CLASS_encoded": "Avg Fire Size Class"},
        title="Wildfire Size Class Across the USA" )

fig.show()
```

Similar to the U.S. map above, we wanted to create a map that would portray how the fire sizes vary by latitude and longitude. We ensured the color choices for the legend would align with reader expectations, and so we chose yellow, orange and red to portray the increasing fire sizes.

[59]:
```
copy["Fire_Lon"]
```

[59]:
```
0        -92.34605
1        -92.34605
2        -92.34605
3        -92.34605
4        -92.34605
            …
19487   -85.523808
19488   -85.523808
19489   -85.523808
19490   -85.523808
19491   -85.523808
Name: Fire_Lon, Length: 19492, dtype: object
```

[76]:
```
import plotly.express as px

cols = ['FIRE_SIZE', 'Fire_Lon', 'Fire_Lat', 'FIRE_SIZE_CLASS_encoded']
copy = final_df.copy()
copy = copy[cols]

# Bubble scaling
size_scale = 10
copy['BUBBLE_SIZE'] = copy['FIRE_SIZE'] / copy['FIRE_SIZE'].max() * size_scale␣
 ↪+ 4
copy['BUBBLE_SIZE'] = copy['BUBBLE_SIZE'].astype(float)
copy['FIRE_SIZE_CLASS_encoded'] = copy['FIRE_SIZE_CLASS_encoded'].astype(str) #␣
 ↪Convert to string for bubble coloring

color_map = {
    '0': '#FFFF00',
    '1': '#FFC300',
    '2': '#FF5733',
    '3': '#C70039',
    '4': '#900C3F'
}

fig = px.scatter(
    copy,
    x='Fire_Lon',
    y='Fire_Lat',
    color='FIRE_SIZE_CLASS_encoded',
    opacity=0.7,
    size=copy['BUBBLE_SIZE'],
    title="Fire Intensity Map",
```

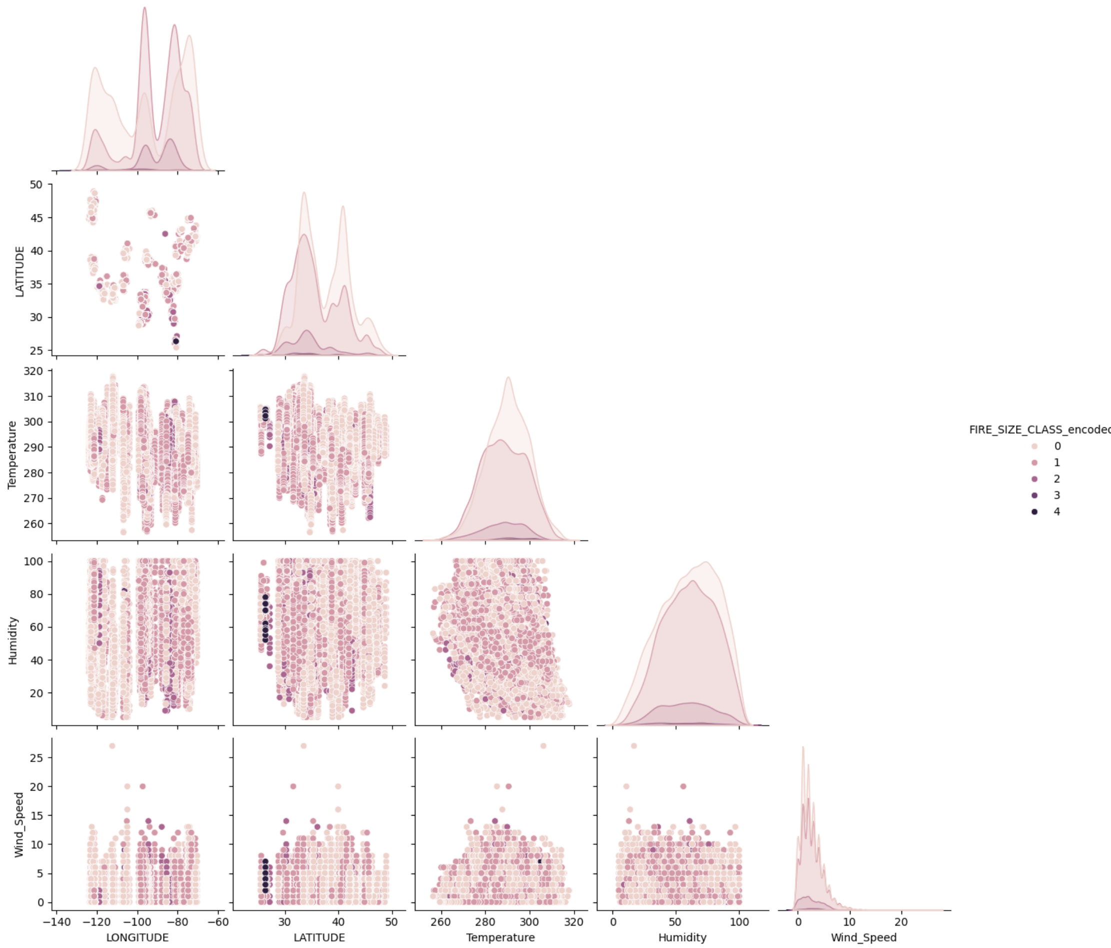
```
    category_orders={"FIRE_SIZE_CLASS_encoded": ["0", "1", "2"]},
    color_discrete_map=color_map,
    hover_data={"FIRE_SIZE": True, "Fire_Lon": True, "Fire_Lat": True,␣
↪"BUBBLE_SIZE": False}  # Modify tooltip legend names
)

fig.update_traces(
    hovertemplate="<b>Fire Size = %{customdata[0]}</b><br>" +
                  "Fire Lon: %{x}<br>" +
                  "Fire Lat: %{y}<br>"
)

fig.update_layout(legend_title_text="Fire Size Class")
fig.show()
```

[77]:
```
final_df.columns
```

[77]:
```
Index(['OBJECTID', 'FOD_ID', 'FPA_ID', 'SOURCE_SYSTEM_TYPE', 'SOURCE_SYSTEM',
       'NWCG_REPORTING_AGENCY', 'NWCG_REPORTING_UNIT_ID',
       'NWCG_REPORTING_UNIT_NAME', 'SOURCE_REPORTING_UNIT',
       'SOURCE_REPORTING_UNIT_NAME', 'LOCAL_FIRE_REPORT_ID',
       'LOCAL_INCIDENT_ID', 'FIRE_CODE', 'FIRE_NAME',
       'ICS_209_INCIDENT_NUMBER', 'ICS_209_NAME', 'MTBS_ID', 'MTBS_FIRE_NAME',
       'COMPLEX_NAME', 'FIRE_YEAR', 'DISCOVERY_DATE', 'DISCOVERY_DOY',
       'DISCOVERY_TIME', 'STAT_CAUSE_CODE', 'STAT_CAUSE_DESCR', 'CONT_DATE',
       'CONT_DOY', 'CONT_TIME', 'FIRE_SIZE', 'FIRE_SIZE_CLASS', 'LATITUDE',
       'LONGITUDE', 'OWNER_CODE', 'OWNER_DESCR', 'STATE', 'COUNTY',
       'FIPS_CODE', 'FIPS_NAME', 'Shape', 'Fire_Lat', 'Fire_Lon', 'City',
       'City_Lat', 'City_Lon', 'Distance_Miles', 'discovery_date', 'Datetime',
       'City', 'Humidity', 'Wind_Direction', 'Temperature', 'Pressure',
       'Wind_Speed', 'FIRE_SIZE_CLASS_encoded'],
      dtype='object')
```

[78]:
```
copy = final_df[["FIRE_SIZE_CLASS_encoded", "LONGITUDE", "LATITUDE",␣
↪"Temperature", "Humidity", "Wind_Speed"]]
plt.figure(figsize=(8, 2))
sns.pairplot(copy, hue="FIRE_SIZE_CLASS_encoded", diag_kind="kde", corner=True)
plt.show()
```

```
<Figure size 800x200 with 0 Axes>
```



[79]:
```
missing_values = final_df.isnull().sum()
missing_percentage = (final_df.isnull().sum() / len(final_df)) * 100

missing_df = pd.DataFrame({'Missing Values': missing_values, 'Percentage (%)':␣
↪missing_percentage})
missing_df = missing_df[missing_df['Missing Values'] > 0].
↪sort_values(by='Missing Values', ascending=False)
print(missing_df)
#print(missing_df.dtypes)
```

```
                       Missing Values  Percentage (%)
MTBS_ID                         19492      100.000000
MTBS_FIRE_NAME                  19492      100.000000
COMPLEX_NAME                    19467       99.871742
ICS_209_INCIDENT_NUMBER         19292       98.973938
ICS_209_NAME                    19292       98.973938
FIRE_CODE                       17570       90.139544
LOCAL_FIRE_REPORT_ID            17531       89.939462
```

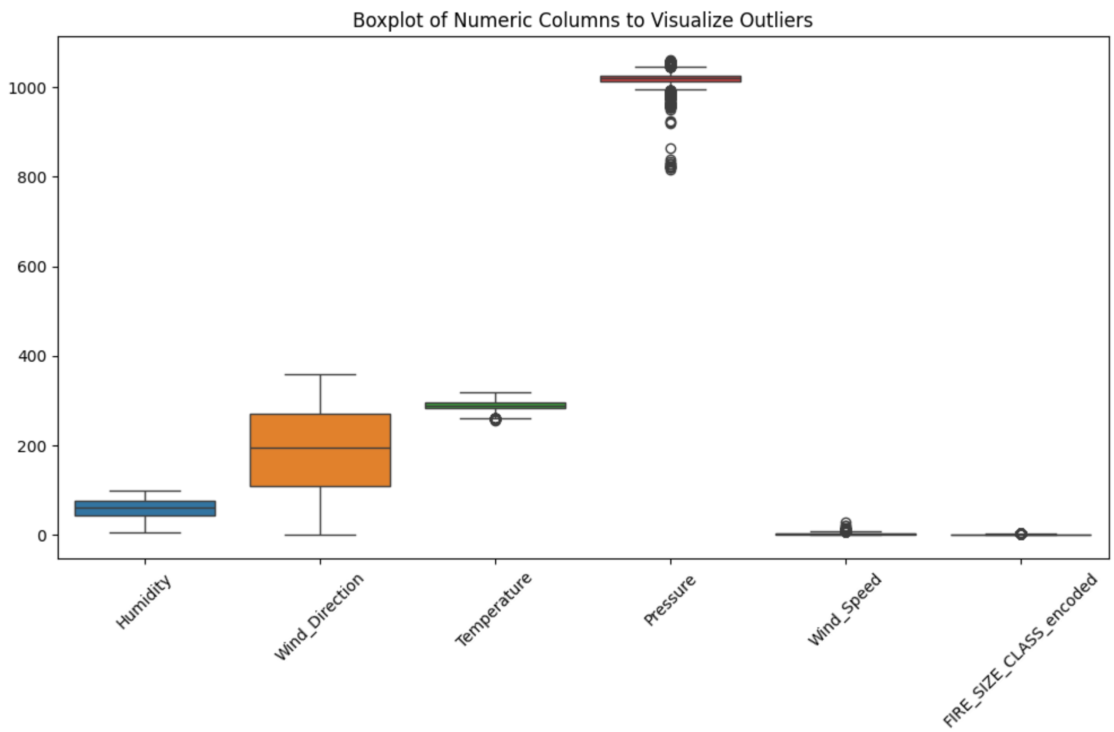
```
CONT_TIME                        6472      33.203365
CONT_DATE                        5154      26.441617
CONT_DOY                         5154      26.441617
DISCOVERY_TIME                   4857      24.917915
FIRE_NAME                        4234      21.721732
COUNTY                           1316       6.751488
FIPS_CODE                        1316       6.751488
FIPS_NAME                        1316       6.751488
LOCAL_INCIDENT_ID                 829       4.253027
```

[80]:
```
#confirming that no missing values are present in the numeric columns

numeric_cols = pd.DataFrame(final_df.select_dtypes(include=[np.number]).columns)
missing_values = numeric_cols.isnull().sum()
missing_percentage = (numeric_cols.isnull().sum() / len(numeric_cols)) * 100

missing_df = pd.DataFrame({'Missing Values': missing_values, 'Percentage (%)':␣
↪missing_percentage})
missing_df = missing_df[missing_df['Missing Values'] > 0].
↪sort_values(by='Missing Values', ascending=False)
missing_df
```

[80]:
```
Empty DataFrame
Columns: [Missing Values, Percentage (%)]
Index: []
```

[82]:
```
from scipy.stats import zscore

numeric_cols = final_df.select_dtypes(include=[np.number]).columns.
↪drop('City_Lat').drop('City_Lon')
z_scores = np.abs(final_df[numeric_cols].apply(zscore))

outlier_counts = (z_scores > 3).sum()
outlier_counts.drop('FIRE_SIZE_CLASS_encoded', inplace=True)
print(outlier_counts)
plt.figure(figsize=(12, 6))
sns.boxplot(data=final_df[numeric_cols])
plt.xticks(rotation=45)
plt.title("Boxplot of Numeric Columns to Visualize Outliers")
plt.show()
```

```
Humidity           0
Wind_Direction     0
Temperature       11
Pressure         167
Wind_Speed       254
dtype: int64
```


Boxplot of Numeric Columns to Visualize Outliers
1000
800
600
400
200
0
Humidity
Wind_Direction
Temperature
Pressure
Wind_Speed
FIRE_SIZE_CLASS_encoded

```
[ ]:
```