fireforce

April 22, 2024

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```python
fire = pd.read_csv("./data/ca_daily_fire_2000_2021-v2.csv")
fire_filt = fire[(fire["year"] >= 2011) & (fire["year"] <= 2014) ]
fire_filt.head()
```

[2]:
```
    latitude  longitude    acq_date satellite instrument   frp  type  \
0    32.4646  -114.6906  2011-06-29     Terra      MODIS  91.1     0
1    32.4768  -114.6785  2011-10-11     Terra      MODIS  96.2     0
2    32.4937  -114.7856  2013-02-06     Terra      MODIS  26.9     0
5    32.5006  -114.7917  2013-02-06     Terra      MODIS  45.0     0
13   32.5167  -114.7978  2011-11-05     Terra      MODIS   8.2     0

    bright_t31  confidence  year  month
0        315.7          84  2011      6
1        313.7         100  2011     10
2        296.1          65  2013      2
5        296.4          74  2013      2
13       295.7          65  2011     11
```

[3]:
```python
pm = pd.read_csv("./data/Daily_Census_Tract-Level_PM2.
 ↪5_Concentrations__2011-2014_20240413.csv")
pm.head()
```

[3]:
```
   year       date  statefips  countyfips       ctfips  latitude  longitude  \
0  2011  30DEC2011         48       48399  48399950100  31.96861  -99.99100
1  2011  30DEC2011         48       48399  48399950200  31.95574  -99.96764
2  2011  30DEC2011         48       48399  48399950500  31.65529 -100.05925
3  2011  30DEC2011         48       48399  48399950600  31.76387  -99.89893
4  2011  30DEC2011         48       48401  48401950100  32.31673  -94.60574

   ds_pm_pred  ds_pm_stdd
0    7.590561    5.439812
1    7.660033    5.666294
2    7.355021    5.490203
3    7.436393    5.247210
```

```
4   11.107991    6.297006
```

[4]:
```
print(fire_filt.columns)
print(pm.columns)
```

```
Index(['latitude', 'longitude', 'acq_date', 'satellite', 'instrument', 'frp',
       'type', 'bright_t31', 'confidence', 'year', 'month'],
      dtype='object')
Index(['year', 'date', 'statefips', 'countyfips', 'ctfips', 'latitude',
       'longitude', 'ds_pm_pred', 'ds_pm_stdd'],
      dtype='object')
```

fire['StartedMonth'] = [x.month for x in wf['Started']] monthly_count = wf.groupby(["ArchiveYear","StartedMonth"])['AcresBurned'].count().reset_index() monthly_count.rename(columns={"AcresBurned": "WildfireCount"}, inplace=True) monthly_count

[5]:
```
precision = 3
fire['rounded_latitude'] = fire['latitude'].round(precision)
fire['rounded_longitude'] = fire['longitude'].round(precision)
pm['rounded_latitude'] = pm['latitude'].round(precision)
pm['rounded_longitude'] = pm['longitude'].round(precision)
```

[6]:
```
fire_monthly_count = fire_filt.groupby(["year","month"])['frp'].count().
 ↪reset_index()
fire_monthly_count.rename(columns={"frp": "Wildfires"}, inplace=True)
fire_monthly_count.head()
```

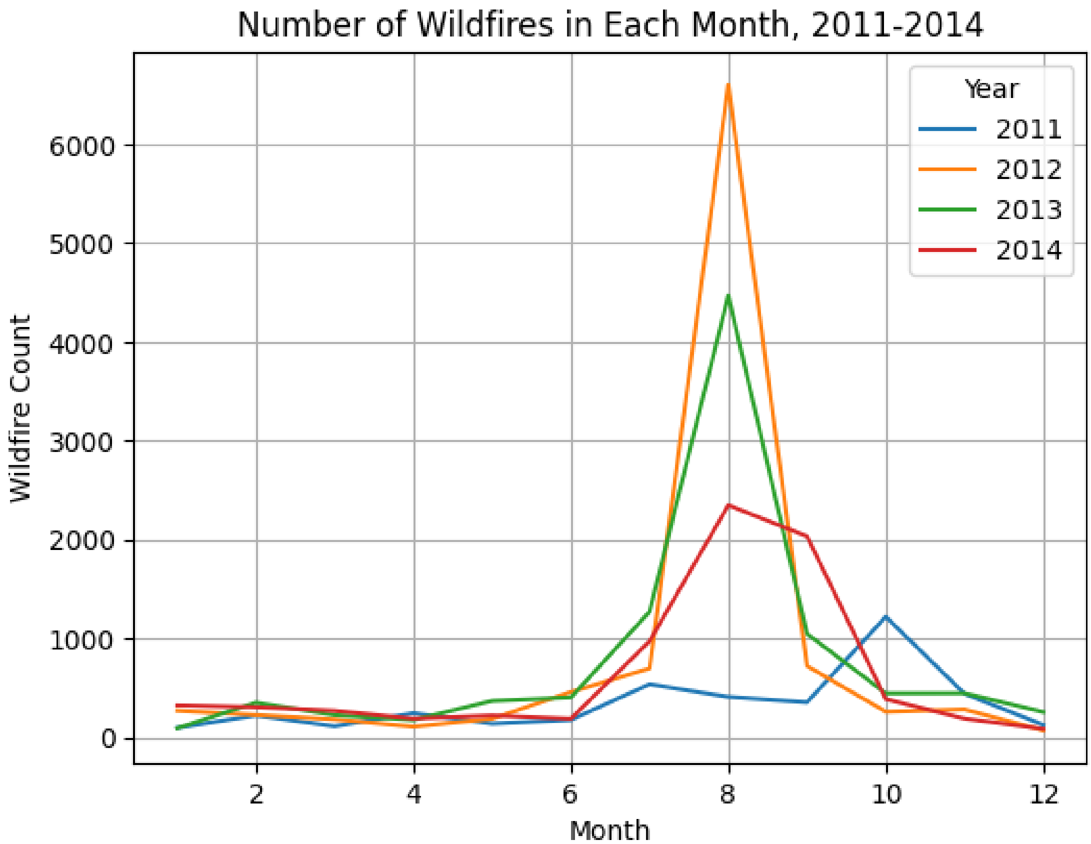
[6]:

	year	month	Wildfires
0	2011	1	95
1	2011	2	220
2	2011	3	111
3	2011	4	243
4	2011	5	137

[7]:
```
fire_monthly_count['month'] = pd.Categorical(fire_monthly_count['month'],␣
 ↪categories=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
                                        ordered=True)

for year in fire_monthly_count['year'].unique():
    subset = fire_monthly_count[fire_monthly_count['year'] == year]
    plt.plot(subset['month'], subset['Wildfires'], label=str(year))

plt.title('Number of Wildfires in Each Month, 2011-2014')
plt.xlabel('Month')
plt.ylabel('Wildfire Count')
plt.legend(title='Year', loc='upper right')
plt.grid(True)
```



When trying to predict PM2.5 concentrations using wildfires we planned on using the amount of wildfires each month. We inferred that if a month had more wildfires, then that month would have a higher concentration of PM2.5. For the number of wildfires, we see a sharp increase in July and August. This seems to be a trend beacuse it happens every year for the data we have.

[8]:
```
pm_cali = pm[pm["statefips"]== 6]
pm_cali['date_parsed'] = pd.to_datetime(pm_cali['date'], format='%d%b%Y')
pm_cali['month'] = pm_cali['date_parsed'].dt.month
pm_cali.head()
```

```
/var/folders/bb/bc8ww5jx0kgdhn7fznlq1nqc0000gn/T/ipykernel_30895/2217055089.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  pm_cali['date_parsed'] = pd.to_datetime(pm_cali['date'], format='%d%b%Y')
/var/folders/bb/bc8ww5jx0kgdhn7fznlq1nqc0000gn/T/ipykernel_30895/2217055089.py:3
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  pm_cali['month'] = pm_cali['date_parsed'].dt.month
```

[8]:
```
       year       date  statefips  countyfips      ctfips  latitude  \
10645  2011  31DEC2011          6        6001  6001400100  37.86754
10646  2011  31DEC2011          6        6001  6001400200  37.84817
10647  2011  31DEC2011          6        6001  6001400300  37.84056
10648  2011  31DEC2011          6        6001  6001400400  37.84801
10649  2011  31DEC2011          6        6001  6001400500  37.84853

       longitude  ds_pm_pred  ds_pm_stdd  rounded_latitude  rounded_longitude  \
10645 -122.23181    8.143539    4.009139            37.868           -122.232
10646 -122.24948    8.116514    3.971881            37.848           -122.249
10647 -122.25442    8.083649    3.933703            37.841           -122.254
10648 -122.25752    8.180107    4.215300            37.848           -122.258
10649 -122.26480    8.125882    4.103174            37.849           -122.265

      date_parsed  month
10645  2011-12-31     12
10646  2011-12-31     12
10647  2011-12-31     12
10648  2011-12-31     12
10649  2011-12-31     12
```

[9]:
```
monthly_pm_cali = pm_cali.groupby(["year","month"])['ds_pm_pred'].mean().
 ↪reset_index()
monthly_pm_cali.rename(columns={"ds_pm_pred": "pm_level"}, inplace=True)
monthly_pm_cali.head()
```

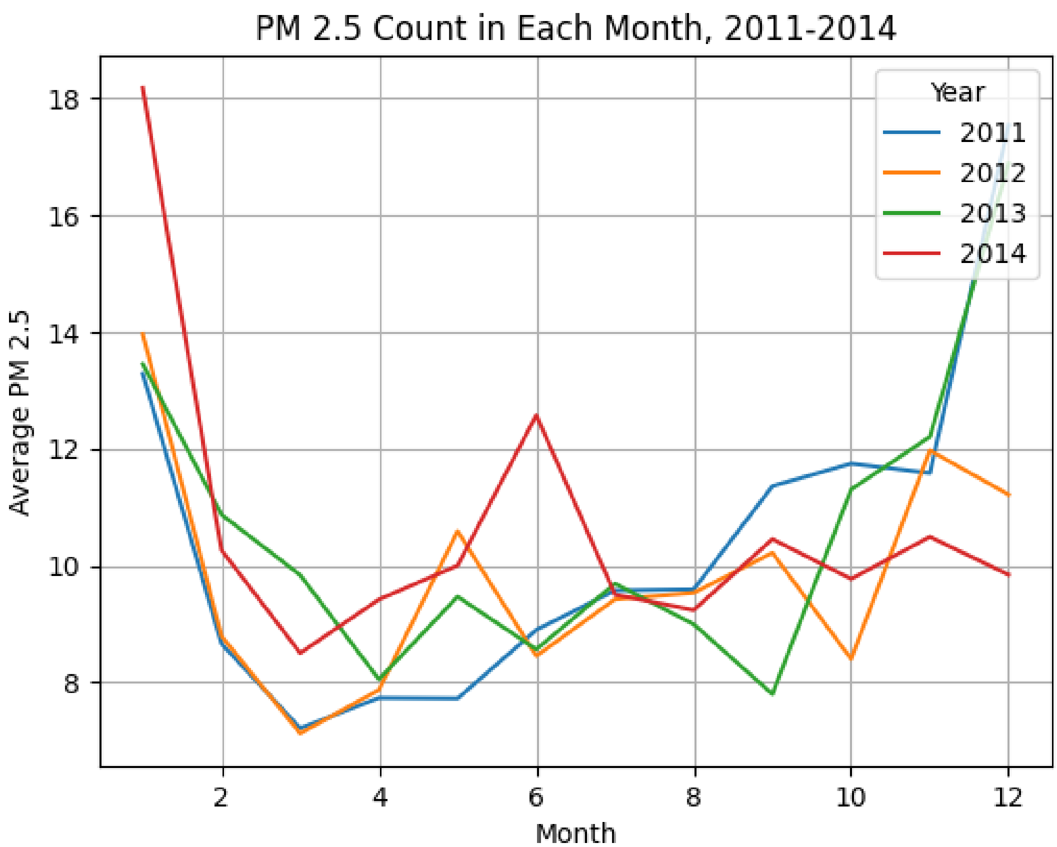
[9]:
```
   year  month   pm_level
0  2011      1  13.283995
1  2011      2   8.666981
2  2011      3   7.220850
3  2011      4   7.737372
4  2011      5   7.729165
```

[10]:
```
monthly_pm_cali['month'] = pd.Categorical(monthly_pm_cali['month'],␣
 ↪categories=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], ordered=True)

for year in monthly_pm_cali['year'].unique():
    subset = monthly_pm_cali[monthly_pm_cali['year'] == year]
    plt.plot(subset['month'], subset['pm_level'], label=str(year))

plt.title('PM 2.5 Count in Each Month, 2011-2014')
plt.xlabel('Month')
```

```
plt.ylabel('Average PM 2.5')
plt.legend(title='Year', loc='upper right')
plt.grid(True)
```



This is the second numerical plot for wildfires. We see an increase in the average PM2.5 for the months November-January. There seems to be a minimum in March and April for most years. This seems too uncorrelated with the months from the wildfire dataset so this feature has to be adapted more for it to be more useful for a GLM. Aggregating the data this way may not have been the best way since wildifres affect PM2.5 levels locally.

[11]:
```
import geopandas as gpd
import folium
from folium.plugins import MarkerCluster

counties = gpd.read_file("./data/cb_2018_06_tract_500k.shp")

# Convert fire_filt DataFrame to a GeoDataFrame
fire_geo = gpd.GeoDataFrame(fire_filt, geometry=gpd.
 ↪points_from_xy(fire_filt['longitude'], fire_filt['latitude']))
```

```python
# Set the CRS of fire_geo to match the CRS of counties
fire_geo.crs = counties.crs

# Perform spatial join to assign county to each wildfire location
fire_county = gpd.sjoin(fire_geo, counties, op='within')

# Aggregate wildfire counts by county
fire_county_counts = fire_county.groupby('COUNTYFP').size().
↪reset_index(name='wildfire_count')

pm_geo = gpd.GeoDataFrame(pm_cali, geometry=gpd.
↪points_from_xy(pm_cali['longitude'], pm_cali['latitude']))

# Set the CRS of pm_geo to match the CRS of counties
pm_geo.crs = counties.crs

pm_county = gpd.sjoin(pm_geo, counties, op='within')
pm_county_avg = pm_county.groupby('COUNTYFP')['ds_pm_pred'].mean().reset_index()

# Merge wildfire counts and PM2.5 averages with the county shapefile
counties_merged = counties.merge(fire_county_counts, on='COUNTYFP', how='left')
counties_merged = counties_merged.merge(pm_county_avg, on='COUNTYFP',␣
↪how='left')
```

```
/Users/otaira/.pyenv/versions/3.12.2/envs/emberalert/lib/python3.12/site-
packages/IPython/core/interactiveshell.py:3448: FutureWarning: The `op`
parameter is deprecated and will be removed in a future release. Please use the
`predicate` parameter instead.
  if await self.run_code(code, result, async_=asy):
/Users/otaira/.pyenv/versions/3.12.2/envs/emberalert/lib/python3.12/site-
packages/IPython/core/interactiveshell.py:3448: FutureWarning: The `op`
parameter is deprecated and will be removed in a future release. Please use the
`predicate` parameter instead.
  if await self.run_code(code, result, async_=asy):
```

[12]:
```python
counties_merged
```

[12]:

	STATEFP	COUNTYFP	TRACTCE	AFFGEOID	GEOID	NAME	\
0	06	009	000300	1400000US06009000300	06009000300	3	
1	06	011	000300	1400000US06011000300	06011000300	3	
2	06	013	303102	1400000US06013303102	06013303102	3031.02	
3	06	013	303202	1400000US06013303202	06013303202	3032.02	
4	06	013	303203	1400000US06013303203	06013303203	3032.03	
…	…	…	…	…	…	…	
8036	06	075	022902	1400000US06075022902	06075022902	229.02	
8037	06	065	044804	1400000US06065044804	06065044804	448.04	
8038	06	099	003300	1400000US06099003300	06099003300	33	

```
8039      06      037  124400  1400000US06037124400  06037124400     1244
8040      06      107  003901  1400000US06107003901  06107003901    39.01

     LSAD      ALAND    AWATER  \
0      CT  457009794    394122
1      CT  952744514    195376
2      CT    6507019         0
3      CT    3725528         0
4      CT    6354210         0
…     …          …        …
8036   CT     161833         0
8037   CT    2374766    248057
8038   CT  640784444   2596432
8039   CT     961439     14163
8040   CT    4993183     25643

                                               geometry  wildfire_count  \
0     POLYGON ((-120.76399 38.21389, -120.76197 38.2…            94.0
1     POLYGON ((-122.50006 39.12232, -122.50022 39.1…          1235.0
2     POLYGON ((-121.72937 37.96884, -121.71409 37.9…            51.0
3     POLYGON ((-121.72346 37.96161, -121.71672 37.9…            51.0
4     POLYGON ((-121.74486 37.95681, -121.74425 37.9…            51.0
…                                                   …              …
8036  POLYGON ((-122.41205 37.75423, -122.40925 37.7…             NaN
8037  POLYGON ((-116.51068 33.80502, -116.51069 33.8…           579.0
8038  POLYGON ((-121.48677 37.47565, -121.48341 37.4…           115.0
8039  POLYGON ((-118.41379 34.17940, -118.41160 34.1…           325.0
8040  POLYGON ((-119.00850 36.07658, -118.99978 36.0…           805.0

      ds_pm_pred
0       7.426730
1       6.937214
2       9.460271
3       9.460271
4       9.460271
…            …
8036    9.142923
8037   10.201348
8038   10.480116
8039   11.889706
8040   13.587403

[8041 rows x 12 columns]
```

[13]:
```
# Create a base map centered on California
california_map = folium.Map(location=[37.7749, -122.4194], zoom_start=6)
```

```
folium.Choropleth(
    geo_data=counties_merged,
    name='PM2.5 Levels',
    data=counties_merged,
    columns=['COUNTYFP', 'ds_pm_pred'],
    key_on='feature.properties.COUNTYFP',
    fill_color='YlOrRd',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Average PM2.5 Level'
).add_to(california_map)

marker_cluster = MarkerCluster().add_to(california_map)

# Add wildfire locations to the marker cluster
for idx, row in counties_merged.iterrows():
    if row['wildfire_count'] > 0:
        folium.Marker(location=[row.geometry.centroid.y, row.geometry.centroid.
 ↪x],
                      popup=f"County: {row['COUNTYFP']}, Wildfires:␣
 ↪{row['wildfire_count']}").add_to(marker_cluster)


# Display the map
california_map
```

[13]:
```
<folium.folium.Map at 0x16a5f2d80>
```

To better see if wildfires affect PM2.5 concenetrations locally, we categorized the PM2.5 and wildfire data by district. The circles represent the number of wildfires. The markers just show the counties they represent. The gradient corresponds to PM2.5 levels. We see a higher count of wildfires in areas that have a higher count of PM2.5. It is apparent though that the high PM2.5 levels are found in places that are very populated. Nevertheless, there does seem to be some correlation even with the data aggregated over all the years. Location is a must have feature in order to line up the data correctly.

[11]:
```
from math import radians, sin, cos, sqrt, atan2

def haversine_distance(lat1, lon1, lat2, lon2):
    # Convert latitude and longitude to radians
    lat1, lon1, lat2, lon2 = map(radians, [lat1, lon1, lat2, lon2])

    # Haversine formula
    dlat = lat2 - lat1
    dlon = lon2 - lon1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * atan2(sqrt(a), sqrt(1-a))
```

```
    distance = 6371 * c  # Earth's radius in kilometers
    return distance
```

[ ]:

```

```

[ ]:

```
def calculate_distance(lat1, lon1, lat2, lon2):
    return haversine((lat1, lon1), (lat2, lon2))

def categorize_distance(distance):
    if distance < 10:
        return '0-10 km'
    elif distance < 50:
        return '10-50 km'
    elif distance < 100:
        return '50-100 km'
    else:
        return '100+ km'

# Calculate distances for each row in the merged dataset
merged_df['distance'] = merged_df.apply(
    lambda row: calculate_distance(row['latitude_x'], row['longitude_x'],␣
 ↪row['latitude_y'], row['longitude_y']),
    axis=1
)

# Categorize distances
merged_df['distance_category'] = merged_df['distance'].
 ↪apply(categorize_distance)

# Remove unnecessary columns for visualization
cleaned_df = merged_df[['date', 'ds_pm_pred', 'distance_category']]

# Group by date and distance category to calculate average PM 2.5 levels
final_df = cleaned_df.groupby(['date', 'distance_category']).mean().
 ↪reset_index()

# Print final prepared DataFrame
print(final_df.head())
```

[ ]:

```
# Assuming df_fire and df_pm are your pre-loaded datasets
# You would need to preprocess these datasets as described above.

# Example of plotting
plt.figure(figsize=(10, 6))
for category in distance_categories:
    subset = df_pm[df_pm['distance_category'] == category]
```

```
    plt.plot(subset['date'], subset['ds_pm_pred'], label=f'Distance: {category}␣
↪km')

plt.xlabel('Date')
plt.ylabel('PM 2.5 Levels')
plt.title('PM 2.5 Levels Over Time by Proximity to Wildfire Events')
plt.legend()
plt.show()
```

[ ]:

```

```