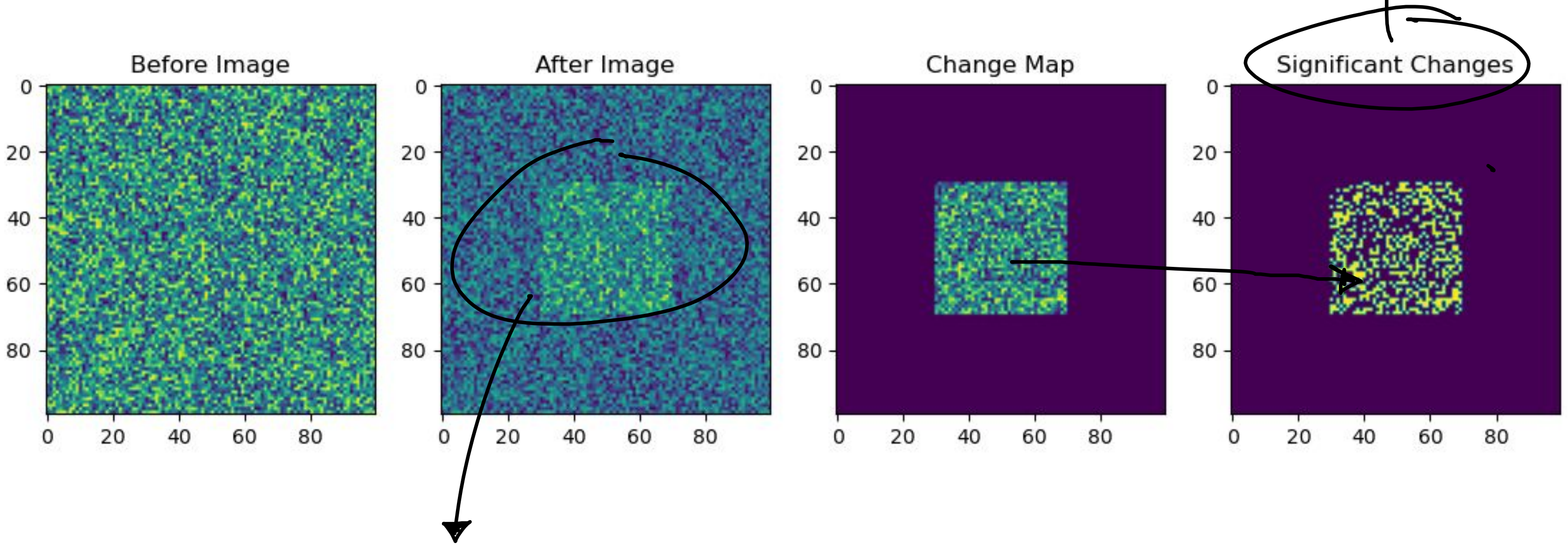
Synthetic SAR Change Detection and Prioritization (SSAR-CDP) Algorithm

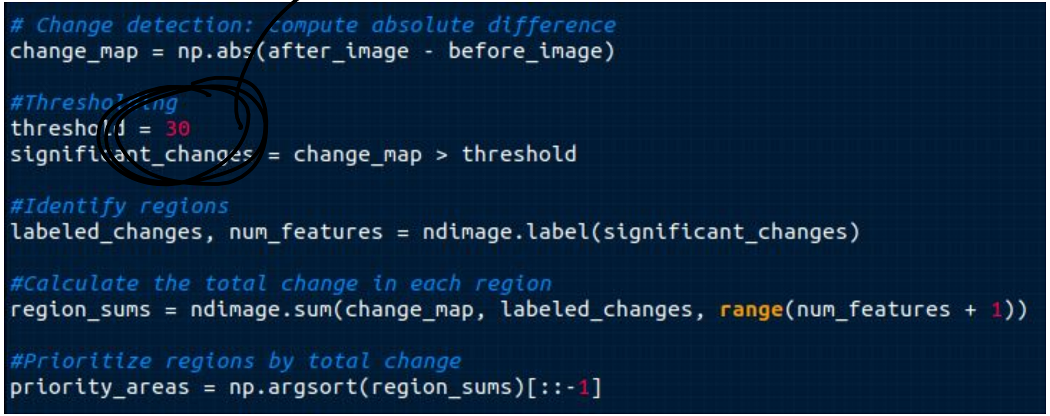


thresholding logic

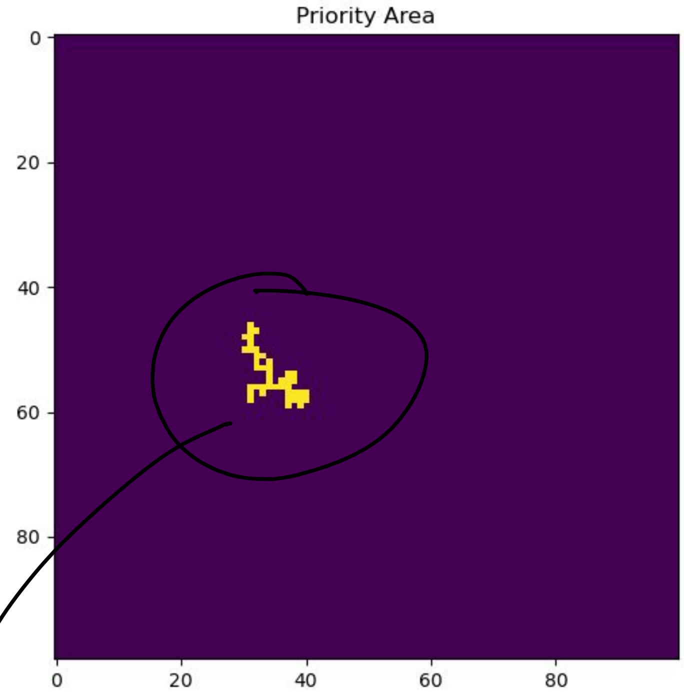
added noise (alias for fire)\

The change detection algorithm is straightforward but effective, making it easy to understand and implement while still providing valuable insights.

(probably use a NN to learn this value???

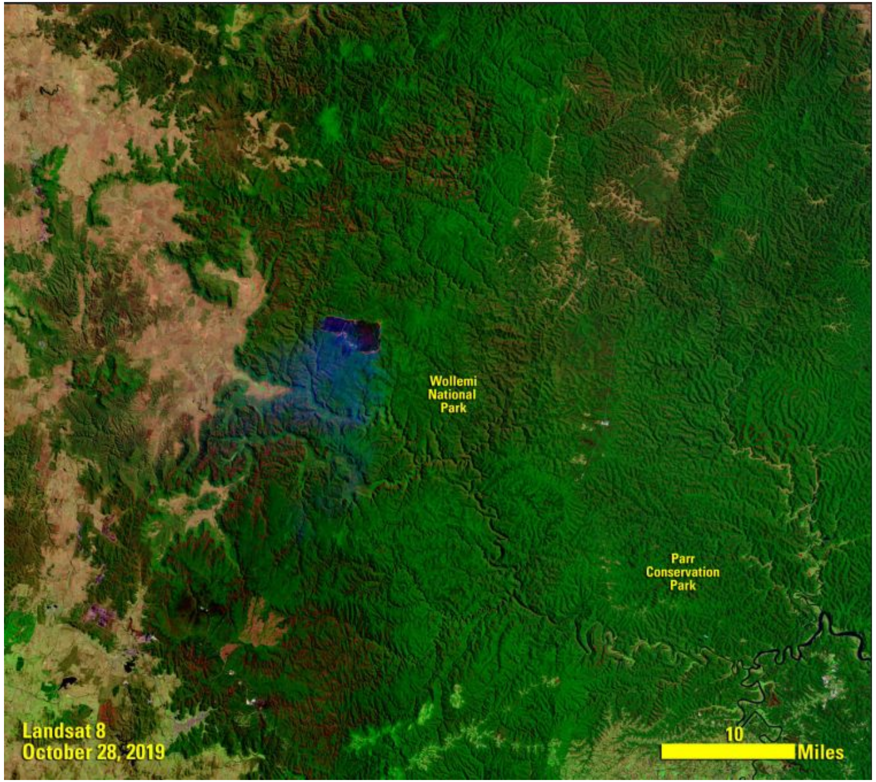


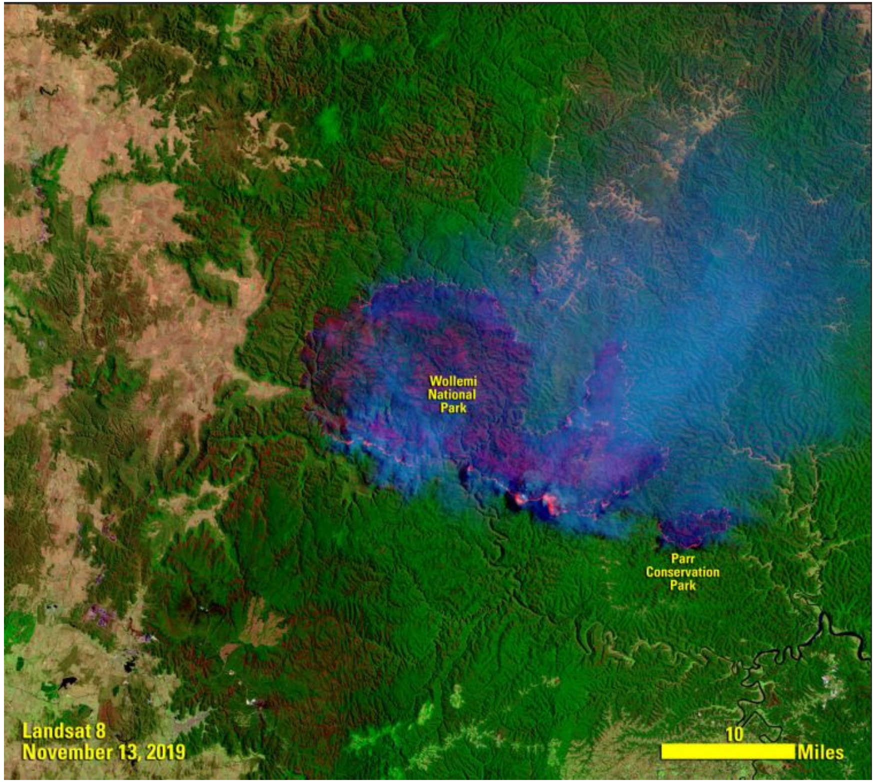
```
# Change detection: compute absolute difference
change_map = np.abs(after_image - before_image)

#Thresholding
threshold = 30
significant_changes = change_map > threshold

#Identify regions
labeled_changes, num_features = ndimage.label(significant_changes)

#Calculate the total change in each region
region_sums = ndimage.sum(change_map, labeled_changes, range(num_features + 1))

#Prioritize regions by total change
priority_areas = np.argsort(region_sums)[::-1]
```



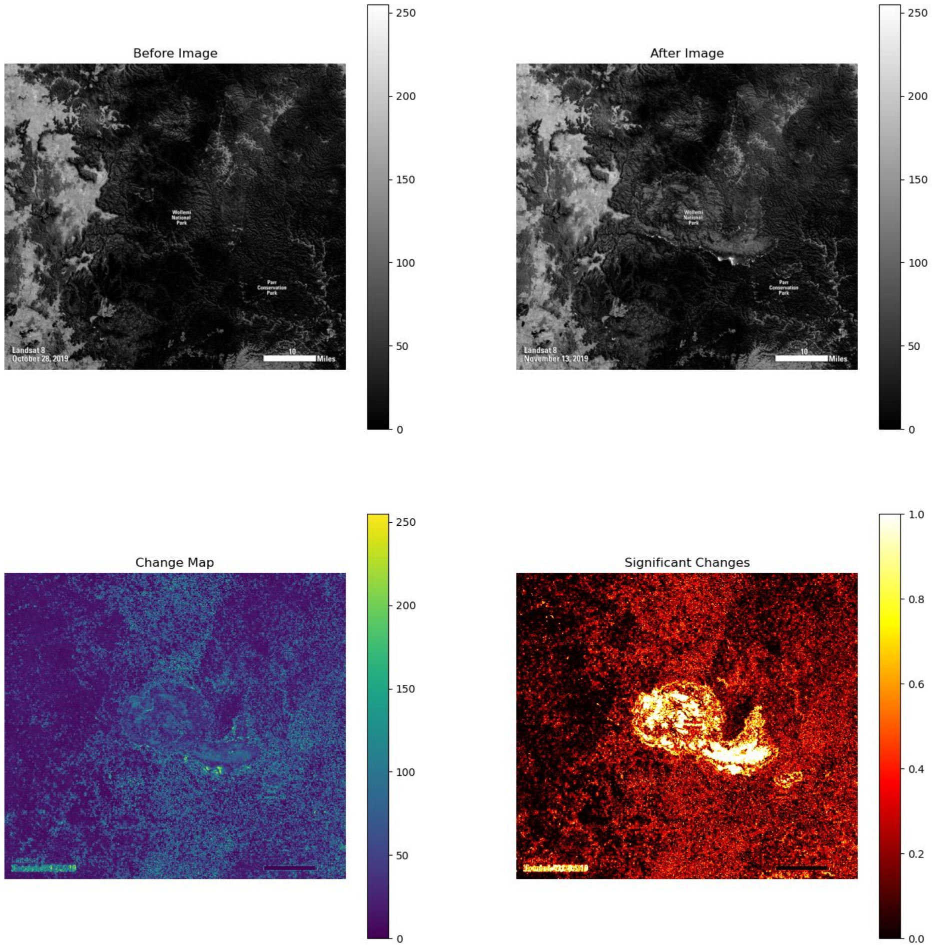
reamap the priority area on the map

Australian bushfire





(October - November)

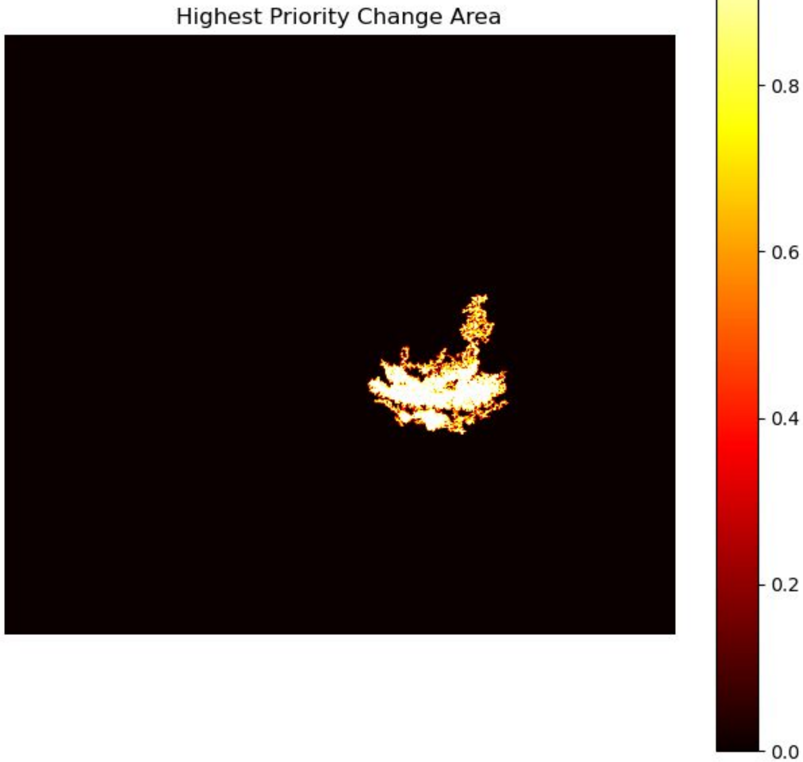


Before Image

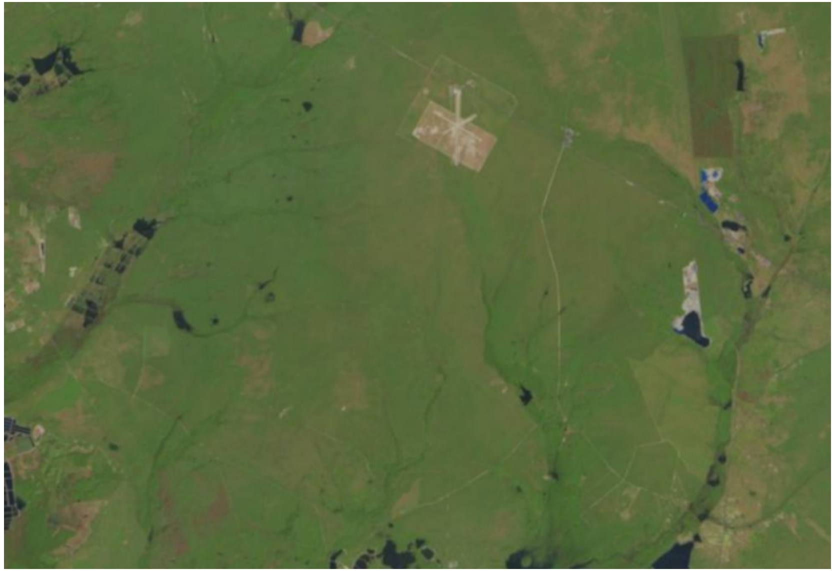
After Image

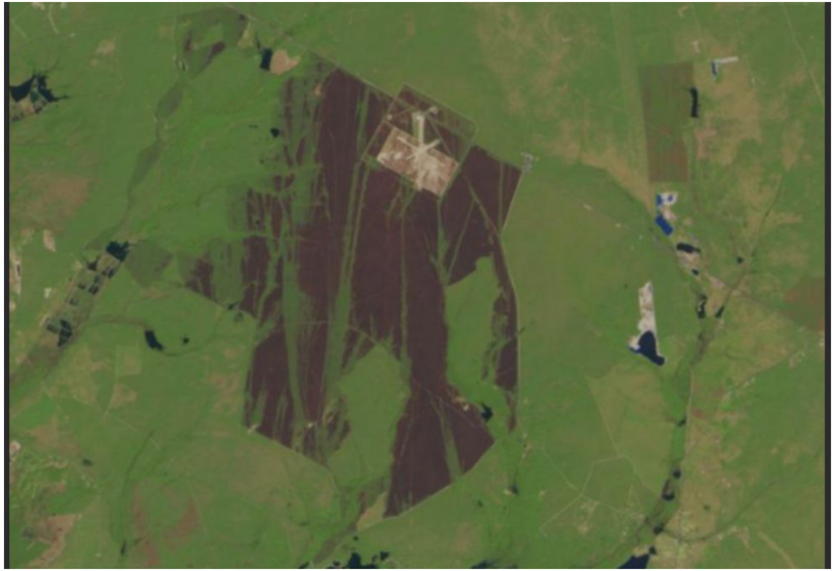
Change Map

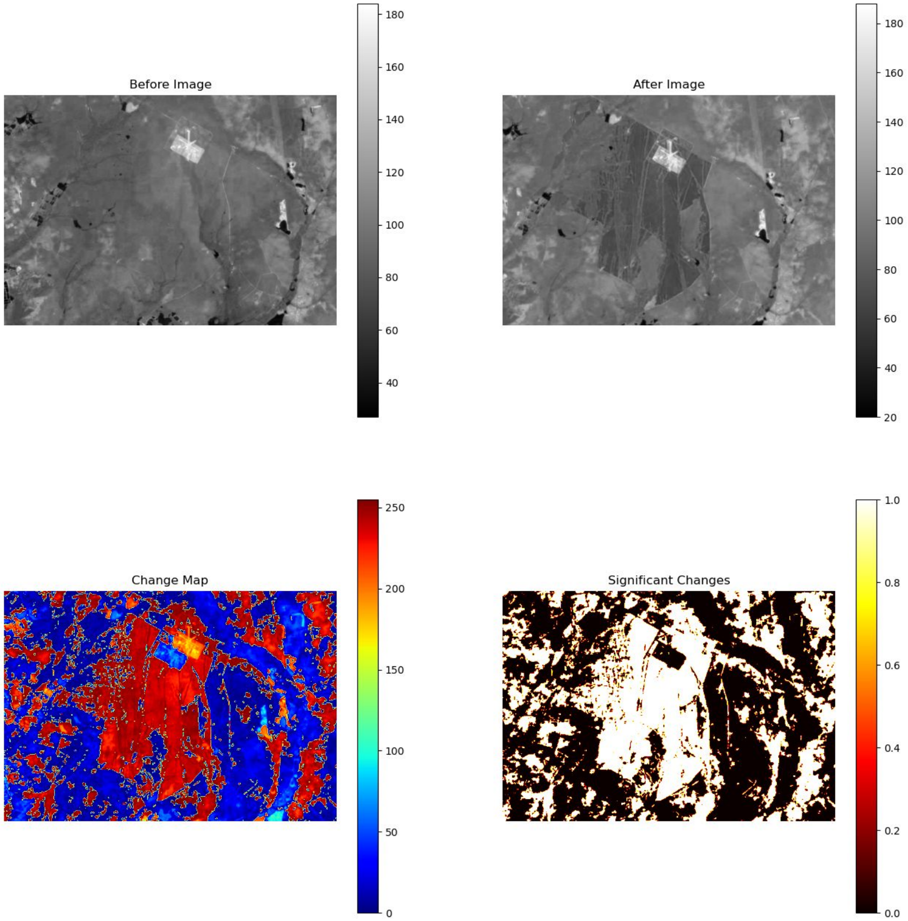
Significant Changes


Highest Priority Change Area

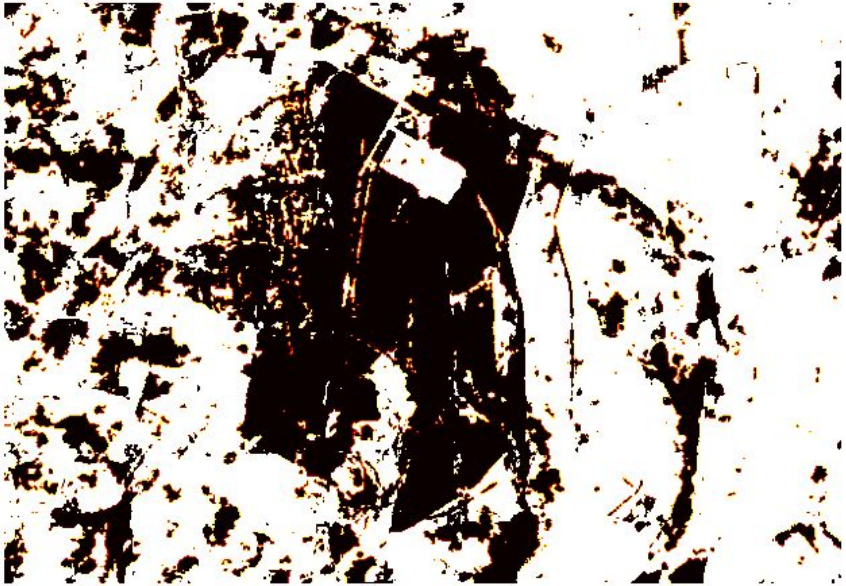
New Jersey forest fire



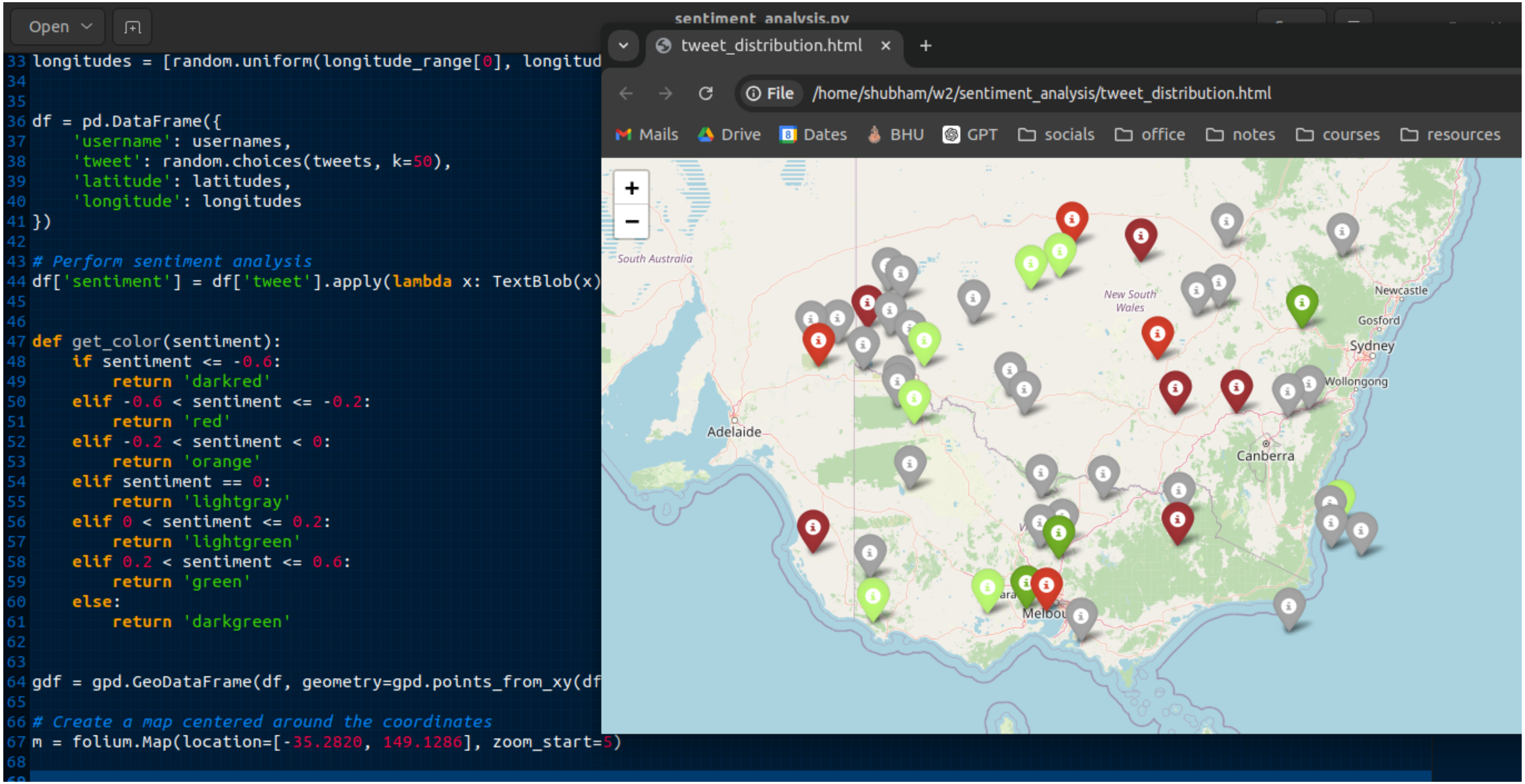



Before Image
180
160
140
120
100
80
60
40
After Image
180
160
140
120
100
80
60
40
20
Change Map
250
200
150
100
50
0
Significant Changes
1.0
0.8
0.6
0.4
0.2
0.0

Highest Priority Change Area



Running a sentiment analysis on the subject region in (REAL TIME)
-> parsing them through an LLM to generate bulletins



```
longitudes = [random.uniform(longitude_range[0], longitud


df = pd.DataFrame({
    'username': usernames,
    'tweet': random.choices(tweets, k=50),
    'latitude': latitudes,
    'longitude': longitudes
})

# Perform sentiment analysis
df['sentiment'] = df['tweet'].apply(lambda x: TextBlob(x)


def get_color(sentiment):
    if sentiment <= -0.6:
        return 'darkred'
    elif -0.6 < sentiment <= -0.2:
        return 'red'
    elif -0.2 < sentiment < 0:
        return 'orange'
    elif sentiment == 0:
        return 'lightgray'
    elif 0 < sentiment <= 0.2:
        return 'lightgreen'
    elif 0.2 < sentiment <= 0.6:
        return 'green'
    else:
        return 'darkgreen'


gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df

# Create a map centered around the coordinates
m = folium.Map(location=[-35.2820, 149.1286], zoom_start=5)

```