Preparing Weather and Proximity Data

In [1]:
```
import pandas as pd
import os
import numpy as np
import geopandas as gpd
import matplotlib.pyplot as plt
import seaborn as sns
import osmnx as ox
import os

import warnings
warnings.filterwarnings("ignore")
```

Import

In [2]:
```
#Importing all count station location data
df_al=pd.read_excel("C:/Users/P-Koirala/OneDrive - Texas A&M Transportation Institu
```

In [3]:
```
#selecting just the necessary columns
df_all=df_al[['stationid', 'Latitude', 'Longitude']]
df_all=df_all.drop_duplicates(subset=['stationid'], keep='first')
df_all.reset_index(drop=True, inplace=True)
```

In [4]:
```
gdf_all=gpd.GeoDataFrame(df_all, geometry=gpd.points_from_xy(df_all.Longitude, df_a
gdf_all.to_crs(epsg=2277, inplace=True)
```

Preparation

In [28]:
```
import os
files= os.listdir('Data/weather/Texas GSOY station data')
weather_files=[f for f in files if f.endswith('.csv')]
dataset=[]
for i in weather_files:
    file=pd.read_csv("Data/weather/Texas GSOY station data/"+i)
    dataset.append(file)
weather_data=pd.concat(dataset)
```

In [41]:
```
#Info on all weather stations in texas (thousands..)
file="Data/weather/Texas GSOY station data/stations_info.txt"
df = pd.read_csv(file, sep='\s+', header=None, usecols=[0,1,2], names=['stationid',
```

In [144...
```
weather_data2=weather_data.merge(df, right_on="stationid", left_on="STATION")
weather_data3=weather_data2[['stationid','DATE','Lon','Lat','PRCP', 'TAVG','TMAX',
weather_data3=weather_data3.rename(columns={"stationid":"weather_station"})
weather_data3['DATE']=(weather_data3['DATE'].astype(str).str[:4]).astype(int)
```

In [196...
```
from math import radians, sin, cos, sqrt, atan2
df_all2=df_all.copy(deep=True)
```

```
weather_data_unq=weather_data3.drop_duplicates(subset = ['weather_station'], keep='
#Haversine formula
def calc_distance(lon1, lat1, lon2, lat2):
    R = 6371  # earth radius in km
    dlon = radians(lon2 - lon1)
    dlat = radians(lat2 - lat1)
    a = sin(dlat/2)**2 + cos(radians(lat1)) * cos(radians(lat2)) * sin(dlon/2)**2
    c = 2 * atan2(sqrt(a), sqrt(1-a))
    distance = R * c
    return distance

for i, row in df_all.iterrows():
    # calculate the distance between this station and all stations in the second da
    distances = []
    for j, row2 in weather_data_unq.iterrows():
        distance = calc_distance(row['Longitude'], row['Latitude'], row2['Lon'], ro
        distances.append((row2['weather_station'], distance, row['stationid']))
    # find the STATIONID in the second dataset with the shortest distance
    min_distance = min(distances, key=lambda x: x[1])
    distances.remove(min_distance)
    min_distance2 = min(distances, key=lambda x: x[1])
    distances.remove(min_distance2)
    min_distance3 = min(distances, key=lambda x: x[1])
    distances.remove(min_distance3)
    min_distance4 = min(distances, key=lambda x: x[1])

    #print(min_distance2)
    # assign the STATIONID to the corresponding row in the first dataset
    df_all2.loc[i, 'weather_station'] = min_distance[0]
    df_all2.loc[i, 'distance(km)'] = min_distance[1]
    df_all2.loc[i, 'weather_station2'] = min_distance2[0]
    df_all2.loc[i, 'distance(km)2'] = min_distance2[1]
    df_all2.loc[i, 'weather_station3'] = min_distance3[0]
    df_all2.loc[i, 'distance(km)3'] = min_distance3[1]
    df_all2.loc[i, 'weather_station4'] = min_distance4[0]
    df_all2.loc[i, 'distance(km)4'] = min_distance4[1]
```

In [198…

```
df_all2.to_csv("Data/Temp/FULL_nearest_weather_stationid_v2.csv")
```

Selecting stations with > 10KM distance

In [207…

```
df_all3=df_all2.copy(deep=True)
df_all3.set_index(df_all3.stationid, drop=True, inplace=True)
df_all3.loc[(df_all3['distance(km)4']>10), 'weather_station4']=np.nan
```

In [368…

```
weather_data4=weather_data3.copy(deep=True)
#weather_data4.set_index(weather_data3.weather_station, drop=True, inplace=True)
```

In [390…

```
df_aa=pd.read_excel("C:/Users/P-Koirala/OneDrive - Texas A&M Transportation Institu
#df_aa=df_aa[['stationid', 'year']]
```

In [362…

```
df_stations=df_aa.merge(df_all3, left_on=df_aa.stationid, right_on=df_all3.stationi
df_stations.rename(columns={'stationid_x':'stationid'}, inplace=True)
df_stations.drop(['stationid_y', 'key_0', 'Latitude', 'Longitude'], axis=1, inplace
```

In [420…

```
a=df_stations.merge(weather_data4, right_on=['weather_station', 'DATE'], left_on=['
b=a.groupby(['stationid','year'],as_index=False )[('DATE',     'PRCP', 'TAVG', 'TM
ws1 = b.rename(columns={c: c+'_1' for c in b.columns if c not in ['stationid', 'yea

a=df_stations.merge(weather_data4, right_on=['weather_station', 'DATE'], left_on=['
b=a.groupby(['stationid','year'],as_index=False )[('DATE',     'PRCP', 'TAVG', 'TM
ws2 = b.rename(columns={c: c+'_2' for c in b.columns if c not in ['stationid', 'yea

a=df_stations.merge(weather_data4, right_on=['weather_station', 'DATE'], left_on=['
b=a.groupby(['stationid','year'],as_index=False )[('DATE',     'PRCP', 'TAVG', 'TM
ws3 = b.rename(columns={c: c+'_3' for c in b.columns if c not in ['stationid', 'yea

a=df_stations.merge(weather_data4, right_on=['weather_station', 'DATE'], left_on=['
b=a.groupby(['stationid','year'],as_index=False )[('DATE',     'PRCP', 'TAVG', 'TM
ws4 = b.rename(columns={c: c+'_4' for c in b.columns if c not in ['stationid', 'yea
stationid      0
year           0
DATE_1       185
PRCP_1       203
TAVG_1       578
TMAX_1       578
TMIN_1       578
AWND_1       603
dtype: int64
stationid      0
year           0
DATE_2       203
PRCP_2       203
TAVG_2       465
TMAX_2       465
TMIN_2       465
AWND_2       480
dtype: int64
stationid      0
year           0
DATE_3       185
PRCP_3       189
TAVG_3       531
TMAX_3       531
TMIN_3       531
AWND_3       566
dtype: int64
stationid      0
year           0
DATE_4       203
PRCP_4       204
TAVG_4       559
TMAX_4       559
TMIN_4       559
AWND_4       568
dtype: int64
```

Filling missing data with another station data

In [446…

```
#PRCP
ws1.loc[ws1.PRCP_1.isna(), 'PRCP_1']= ws2.loc[ws1.PRCP_1.isna(), 'PRCP_2']
ws1.loc[ws1.PRCP_1.isna(), 'PRCP_1']= ws3.loc[ws1.PRCP_1.isna(), 'PRCP_3']
ws1.loc[ws1.PRCP_1.isna(), 'PRCP_1']= ws4.loc[ws1.PRCP_1.isna(), 'PRCP_4']
#TAVG
ws1.loc[ws1.TAVG_1.isna(), 'TAVG_1']= ws2.loc[ws1.TAVG_1.isna(), 'TAVG_2']
ws1.loc[ws1.TAVG_1.isna(), 'TAVG_1']= ws3.loc[ws1.TAVG_1.isna(), 'TAVG_3']
ws1.loc[ws1.TAVG_1.isna(), 'TAVG_1']= ws4.loc[ws1.TAVG_1.isna(), 'TAVG_4']
#TMAX
ws1.loc[ws1.TMAX_1.isna(), 'TMAX_1']= ws2.loc[ws1.TMAX_1.isna(), 'TMAX_2']
ws1.loc[ws1.TMAX_1.isna(), 'TMAX_1']= ws3.loc[ws1.TMAX_1.isna(), 'TMAX_3']
ws1.loc[ws1.TMAX_1.isna(), 'TMAX_1']= ws4.loc[ws1.TMAX_1.isna(), 'TMAX_4']
#TMIN
ws1.loc[ws1.TMIN_1.isna(), 'TMIN_1']= ws2.loc[ws1.TMIN_1.isna(), 'TMIN_2']
ws1.loc[ws1.TMIN_1.isna(), 'TMIN_1']= ws3.loc[ws1.TMIN_1.isna(), 'TMIN_3']
ws1.loc[ws1.TMIN_1.isna(), 'TMIN_1']= ws4.loc[ws1.TMIN_1.isna(), 'TMIN_4']
```

```
#AWND
ws1.loc[ws1.AWND_1.isna(), 'AWND_1']= ws2.loc[ws1.AWND_1.isna(), 'AWND_2']
ws1.loc[ws1.AWND_1.isna(), 'AWND_1']= ws3.loc[ws1.AWND_1.isna(), 'AWND_3']
ws1.loc[ws1.AWND_1.isna(), 'AWND_1']= ws4.loc[ws1.AWND_1.isna(), 'AWND_4']
```

Fill remaining missing values with mean of other stations

In [502…

```
f = {
    'PRCP_1': np.mean,
    'TAVG_1': np.mean,
    'TMAX_1': np.mean,
    'TMIN_1': np.mean,
    'AWND_1': np.mean
}
grouped = ws1.groupby(['stationid']).agg(f) #finding mean for FIllna
merged = ws1.merge(grouped, on='stationid', suffixes=('_ws1', '_grouped'))
# Fill the null values in the merged dataset with the values from 'grouped'
merged['PRCP_1_ws1'].fillna(merged['PRCP_1_grouped'], inplace=True)
merged['TAVG_1_ws1'].fillna(merged['TAVG_1_grouped'], inplace=True)
merged['TMAX_1_ws1'].fillna(merged['TMAX_1_grouped'], inplace=True)
merged['TMIN_1_ws1'].fillna(merged['TMIN_1_grouped'], inplace=True)
merged['AWND_1_ws1'].fillna(merged['AWND_1_grouped'], inplace=True)

# Drop the columns with '_grouped' suffix
merged.drop(['PRCP_1_grouped', 'TAVG_1_grouped', 'TMAX_1_grouped', 'TMIN_1_grouped'
DF=merged.rename(columns={'PRCP_1_ws1':'PRCP',
                        'TAVG_1_ws1':'TAVG',
                        'TMAX_1_ws1':'TMAX',
                        'TMIN_1_ws1':'TMIN',
                        'AWND_1_ws1':'AWND',})
DF.drop(['DATE_1'], axis=1, inplace=True)
```

In [510…

```
DF.stationid.str[0].unique()
```

Out[510]:

```
array(['A', 'C', 'D', 'E', 'F', 'H', 'L', 'P', 'R'], dtype=object)
```

In [511…

```
DF[DF.stationid.str[0]=="A"]=DF[DF.stationid.str[0]=="A"].fillna(DF.mean(axis=0))
DF[DF.stationid.str[0]=="C"]=DF[DF.stationid.str[0]=="C"].fillna(DF.mean(axis=0))
DF[DF.stationid.str[0]=="D"]=DF[DF.stationid.str[0]=="D"].fillna(DF.mean(axis=0))
DF[DF.stationid.str[0]=="E"]=DF[DF.stationid.str[0]=="E"].fillna(DF.mean(axis=0))
DF[DF.stationid.str[0]=="F"]=DF[DF.stationid.str[0]=="F"].fillna(DF.mean(axis=0))
DF[DF.stationid.str[0]=="H"]=DF[DF.stationid.str[0]=="H"].fillna(DF.mean(axis=0))
DF[DF.stationid.str[0]=="L"]=DF[DF.stationid.str[0]=="L"].fillna(DF.mean(axis=0))
DF[DF.stationid.str[0]=="P"]=DF[DF.stationid.str[0]=="P"].fillna(DF.mean(axis=0))
DF[DF.stationid.str[0]=="R"]=DF[DF.stationid.str[0]=="R"].fillna(DF.mean(axis=0))
```

In [512…

```
DF.to_csv("Data/Temp/FULL_nearest_weather_data_missingtreated_v3.csv")
```

Distance to nearest water and campus

In [53]:

```
gdf_all=gpd.GeoDataFrame(df_all, geometry=gpd.points_from_xy(df_all.Longitude, df_a
# gdf_all.to_crs(epsg=2277, inplace=True)
```

In [7]:

```
import osmnx as ox
import pandas as pd

# get water body data using OpenStreetMap
place_name = "Texas"
tags = {"natural": "water"}
water = ox.geometries_from_place(place_name, tags)
water_geometry = water['geometry'].unary_union
```

In [8]:

```
water.to_crs(epsg=2277, inplace=True)
gdf_all.to_crs(epsg=2277, inplace=True)
```

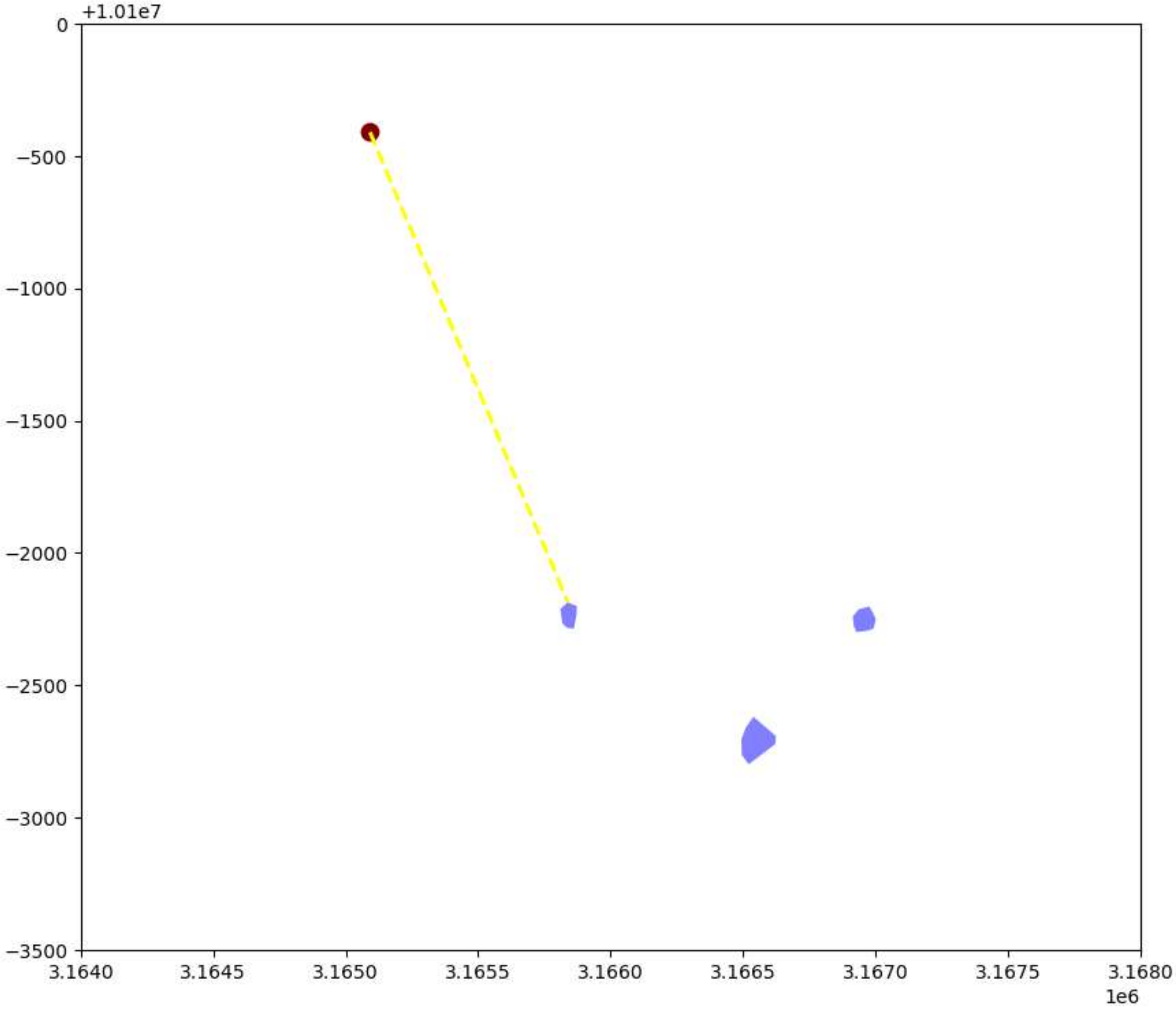
In [9]:

```
from shapely.geometry import Point


# assume your data is in GeoDataFrames called `gdf_all` and `water`
# extract the geometry of the water bodies
water_geometry = water['geometry'].unary_union

# define a function to calculate the minimum distance between a point and the water
def min_distance_to_water(point, water_geometry):
    return point.distance(water_geometry)

# calculate the distance for each point
gdf_all['proximity_water'] = gdf_all.geometry.apply(min_distance_to_water, water_ge
```

In [52]:

```
#VISUALIZE
from shapely.geometry import Polygon, MultiPolygon, LineString

# Find the nearest water body to the station
distances = water['geometry'].apply(lambda x: x.distance(gdf_all.geometry.iloc[0]))
nearest_water = distances.idxmin()
nearest_water_geom = water.loc[nearest_water].geometry

station_point = gdf_all.geometry.iloc[0]

if isinstance(nearest_water_geom, (Polygon, MultiPolygon)):
    nearest_water_geom = nearest_water_geom.boundary

# Calculate the nearest point on the water body to the station
nearest_point = nearest_water_geom.interpolate(nearest_water_geom.project(station_p

# Create a LineString connecting the station and the nearest point on the water bod
line = LineString([station_point, nearest_point])

# Create a GeoDataFrame for the line
line_gdf = gpd.GeoDataFrame(geometry=[line])

# Plot the data
ax = water.plot(color='blue', alpha=0.5, figsize=(10, 10))
gdf_all.iloc[[0]].plot(ax=ax, color='maroon', markersize=80)
line_gdf.plot(ax=ax, color='yellow', linestyle='--', linewidth=2)
# ax.set_xlim(2887000,2892000)
# ax.set_ylim(9735000,9740000)
```

```
ax.set_xlim(3164000,3168000)
ax.set_ylim(10096500,10100000)
plt.show()
```



In [ ]:

```
```

Campus

In [554…

```
gdf_all2=gdf_all.copy(deep=True)
```

In [539…

```
state = 'Texas'
amenity = ['university', 'community_college']

# Query OpenStreetMap to get the universities in Texas
query = f'amenity={amenity} and addr:state={state}'
uni = ox.geometries_from_place(state, tags={'amenity': amenity}, which_result=None)
```

In [540…

```
uni.to_crs(epsg=2277, inplace=True)
```

In [555…

```
uni_geometry = uni['geometry'].unary_union
def min_distanc(point, uni_geometry):
    return point.distance(uni_geometry)
```

```
gdf_all2['distance_uni'] = gdf_all2.geometry.apply(min_distanc, uni_geometry=uni_ge
```

In [558…

```
gdf_all2=gdf_all2.rename(columns={'distance_to_water':'distance_to_water(ft)'})
```

In [559…

```
#gdf_all2.to_csv("Data/Temp/FULL_distance_uni_data_.csv")
```

School

In [ ]:

```
state = 'Texas'
amenity = ['school']

# Query OpenStreetMap to get the universities in Texas
query = f'amenity={amenity} and addr:state={state}'
sco = ox.geometries_from_place(state, tags={'amenity': amenity}, which_result=None)
```

In [ ]:

```
uni.to_crs(epsg=2277, inplace=True)
```

In [ ]:

```
sco_geometry = sco['geometry'].unary_union
def min_distanc(point, sco_geometry):
    return point.distance(sco_geometry)

gdf_all2['proximity_school'] = gdf_all2.geometry.apply(min_distanc, sco_geometry=sc
```

In [ ]:

```
gdf_all2.to_csv("Data/Temp/FULL_proximity_data_v3.csv")
```

In [ ]:

```
#gdf_all2=gdf_all2.rename(columns={'distance_to_water':'distance_to_water(ft)'})
```