Dataset Combination Logic

This notebook is responsible for dataset normalization, and merging the NOAA weather dataset with the forest fire dataset.

```
pip install haversine
```

```
Requirement already satisfied: haversine in /usr/local/lib/python3.7/dist-packages (2.5.1)
```

```
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Data Cleaning and Feature Engineering Utility Functions

```
def count_missing_data(df):
  # Count the number of missing values by feature:
  df_na = df.isna().sum()
  df_na = df_na.drop(df_na[df_na == 0].index).sort_values(ascending=False)
  df_na = (df_na / len(df)) * 100
  missing_data = pd.DataFrame({"Missing Ratio" : df_na})
  display(missing_data)
  return missing_data
```

```
from haversine import haversine
"""
Given two coordinates, the haversine distance is the distance between the points, as the crow-flies
Returns distance in kilometers.
Source: https://www.movable-type.co.uk/scripts/latlong.html
"""
```

```
'\nGiven two coordinates, the haversine distance is the distance between the points, as the crow-flies\nReturns distance in kilometers.\nSource: https://www.movable-type.co.uk/scripts/latlong.html\n'
```

```
import pandas as pd
land_temp_by_state = pd.read_csv("/content/drive/Shareddrives/Data Science 303 Group Project/csv/GlobalLandTemperatures/GlobalLandTemperaturesByState.csv")
```

Dataset One: Average Temperature and Average Temperature Uncertainty of California

```
ca_yearly_average_temp_raw = land_temp_by_state[land_temp_by_state.State == "California"]
ca_yearly_average_temp_raw["dt"] = pd.to_datetime(ca_yearly_average_temp_raw["dt"])

ca_yearly_average_temp = pd.DataFrame()
ca_yearly_average_temp["Year"] = ca_yearly_average_temp_raw["dt"].dt.year
ca_yearly_average_temp["Month"] = ca_yearly_average_temp_raw["dt"].dt.month
ca_yearly_average_temp["AverageTemperature"] = ca_yearly_average_temp_raw["AverageTemperature"]
ca_yearly_average_temp["AverageTemperatureUncertainty"] = ca_yearly_average_temp_raw["AverageTemperatureUncertainty"]

display(ca_yearly_average_temp)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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
```

	Year	Month	AverageTemperature	AverageTemperatureUncertainty
71058	1849	1	5.591	2.405
71059	1849	2	6.941	2.041
71060	1849	3	9.731	2.294
71061	1849	4	12.294	2.861
71062	1849	5	14.417	2.215
...	...	...	...	...
73030	2013	5	17.899	0.228
73031	2013	6	22.513	0.265
73032	2013	7	25.563	0.206
73033	2013	8	23.460	0.369
73034	2013	9	21.924	0.861

Dataset Two: Wildfires in the United States 1992 - 2015

```
fires_raw = pd.read_csv("/content/drive/Shareddrives/Data Science 303 Group Project/csv/us_wildfire_data/fires.csv")
del fires_raw["Shape"] # Deleting this here because the column is causing me issues later and is 99% empty
fires_raw = fires_raw[fires_raw.STATE == "CA"] # Only look at the state of california
fires = pd.DataFrame()
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWarning: Columns (8,10,11,12,13,14,15,16,17,18,35,37) have mixed types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

```
display(fires_raw)
fires_raw.info()
```

	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_FI
0	1	1	FS-1418826	FED	FS-FIRESTAT	FS	USCAPNF	Plumas National Forest	511	Plumas National Forest	
1	2	2	FS-1418827	FED	FS-FIRESTAT	FS	USCAENF	Eldorado National Forest	503	Eldorado National Forest	
2	3	3	FS-1418835	FED	FS-FIRESTAT	FS	USCAENF	Eldorado National Forest	503	Eldorado National Forest	
3	4	4	FS-1418845	FED	FS-FIRESTAT	FS	USCAENF	Eldorado National Forest	503	Eldorado National Forest	
4	5	5	FS-1418847	FED	FS-FIRESTAT	FS	USCAENF	Eldorado National Forest	503	Eldorado National Forest	
...	...	...	...	...	...	...	...	...	...	...	
1880460	1880461	300348363	2015CAIRS29019636	NONFED	ST-CACDF	ST/C&L	USCASHU	Shasta-Trinity Unit	CASHU	Shasta-Trinity Unit	
1880461	1880462	300348373	2015CAIRS29217935	NONFED	ST-CACDF	ST/C&L	USCATCU	Tuolumne-Calaveras Unit	CATCU	Tuolumne-Calaveras Unit	
1880462	1880463	300348375	2015CAIRS28364460	NONFED	ST-CACDF	ST/C&L	USCATCU	Tuolumne-Calaveras Unit	CATCU	Tuolumne-Calaveras Unit	
1880463	1880464	300348377	2015CAIRS29218079	NONFED	ST-CACDF	ST/C&L	USCATCU	Tuolumne-Calaveras Unit	CATCU	Tuolumne-Calaveras Unit	
1880464	1880465	300348399	2015CAIRS26733926	NONFED	ST-CACDF	ST/C&L	USCABDU	San Bernardino Unit	CABDU	CDF - San Bernardino Unit	

189550 rows × 38 columns

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 189550 entries, 0 to 1880464
Data columns (total 38 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   OBJECTID                    189550 non-null  int64  
 1   FOD_ID                      189550 non-null  int64  
 2   FPA_ID                      189550 non-null  object 
 3   SOURCE_SYSTEM_TYPE          189550 non-null  object 
 4   SOURCE_SYSTEM               189550 non-null  object 
 5   NWCG_REPORTING_AGENCY       189550 non-null  object 
 6   NWCG_REPORTING_UNIT_ID      189550 non-null  object 
 7   NWCG_REPORTING_UNIT_NAME    189550 non-null  object 
 8   SOURCE_REPORTING_UNIT       189550 non-null  object 
 9   SOURCE_REPORTING_UNIT_NAME  189550 non-null  object 
 10  LOCAL_FIRE_REPORT_ID        61933 non-null   object 
 11  LOCAL_INCIDENT_ID           127983 non-null  object 
 12  FIRE_CODE                   55522 non-null   object 
 13  FIRE_NAME                   174555 non-null  object 
 14  ICS_209_INCIDENT_NUMBER     2838 non-null    object 
 15  ICS_209_NAME                2838 non-null    object 
 16  MTBS_ID                     1137 non-null    object 
 17  MTBS_FIRE_NAME              1137 non-null    object 
 18  COMPLEX_NAME                927 non-null     object 
 19  FIRE_YEAR                   189550 non-null  int64  
 20  DISCOVERY_DATE              189550 non-null  float64
 21  DISCOVERY_DOY               189550 non-null  int64  
 22  DISCOVERY_TIME              110217 non-null  float64
 23  STAT_CAUSE_CODE             189550 non-null  float64
 24  STAT_CAUSE_DESCR            189550 non-null  object 
 25  CONT_DATE                   91908 non-null   float64
 26  CONT_DOY                    91908 non-null   float64
 27  CONT_TIME                   91198 non-null   float64
 28  FIRE_SIZE                   189550 non-null  float64
 29  FIRE_SIZE_CLASS             189550 non-null  object 
 30  LATITUDE                    189550 non-null  float64
 31  LONGITUDE                   189550 non-null  float64
 32  OWNER_CODE                  189550 non-null  float64
 33  OWNER_DESCR                 189550 non-null  object 
 34  STATE                       189550 non-null  object 
 35  COUNTY                      56221 non-null   object 
 36  FIPS_CODE                   56221 non-null   float64
```

```
# Get the names for each cause code
causes = fires_raw[["STAT_CAUSE_CODE", "STAT_CAUSE_DESCR"]]
code_to_name = {}
for row in causes[:-1].values:
  code_to_name[int(row[0])] = row[1]
print(code_to_name)
```

```
{9: 'Miscellaneous', 1: 'Lightning', 5: 'Debris Burning', 4: 'Campfire', 2: 'Equipment Use', 8: 'Children', 7: 'Arson', 3: 'Smoking', 6: 'Railroad', 10: 'Fireworks', 12: 'Structure', 11: 'Powerline', 13: 'Missing/Undefined'}
```

Dataset Two: Wildfires Cleaning

Dataset Two: Wildfires Drop Missing Values

```
missing = count_missing_data(fires_raw)

drops = missing[missing["Missing Ratio"] > 60].index
print(f"Dropping columns { drops }")
for feature in drops:
  del fires_raw[feature]
```

```

fires_raw.info()
```

	Missing Ratio
COMPLEX_NAME	99.510947
MTBS_FIRE_NAME	99.400158
MTBS_ID	99.400158
ICS_209_NAME	98.502770
ICS_209_INCIDENT_NUMBER	98.502770
FIRE_CODE	70.708520
FIPS_NAME	70.339752
FIPS_CODE	70.339752
COUNTY	70.339752
LOCAL_FIRE_REPORT_ID	67.326299
CONT_TIME	51.887101
CONT_DOY	51.512530
CONT_DATE	51.512530
DISCOVERY_TIME	41.853337
LOCAL_INCIDENT_ID	32.480612
FIRE_NAME	7.910841

```
Dropping columns Index(['COMPLEX_NAME', 'MTBS_FIRE_NAME', 'MTBS_ID', 'ICS_209_NAME',
       'ICS_209_INCIDENT_NUMBER', 'FIRE_CODE', 'FIPS_NAME', 'FIPS_CODE',
       'COUNTY', 'LOCAL_FIRE_REPORT_ID'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
Int64Index: 189550 entries, 0 to 1880464
Data columns (total 28 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   OBJECTID                    189550 non-null  int64
 1   FOD_ID                      189550 non-null  int64
 2   FPA_ID                      189550 non-null  object
 3   SOURCE_SYSTEM_TYPE          189550 non-null  object
 4   SOURCE_SYSTEM               189550 non-null  object
 5   NWCG_REPORTING_AGENCY       189550 non-null  object
 6   NWCG_REPORTING_UNIT_ID      189550 non-null  object
 7   NWCG_REPORTING_UNIT_NAME    189550 non-null  object
 8   SOURCE_REPORTING_UNIT       189550 non-null  object
 9   SOURCE_REPORTING_UNIT_NAME  189550 non-null  object
 10  LOCAL_INCIDENT_ID           127983 non-null  object
 11  FIRE_NAME                   174555 non-null  object
 12  FIRE_YEAR                   189550 non-null  int64
 13  DISCOVERY_DATE              189550 non-null  float64
 14  DISCOVERY_DOY               189550 non-null  int64
 15  DISCOVERY_TIME              110217 non-null  float64
 16  STAT_CAUSE_CODE             189550 non-null  float64
 17  STAT_CAUSE_DESCR            189550 non-null  object
 18  CONT_DATE                   91908 non-null   float64
 19  CONT_DOY                    91908 non-null   float64
 20  CONT_TIME                   91198 non-null   float64
 21  FIRE_SIZE                   189550 non-null  float64
 22  FIRE_SIZE_CLASS             189550 non-null  object
 23  LATITUDE                    189550 non-null  float64
 24  LONGITUDE                   189550 non-null  float64
 25  OWNER_CODE                  189550 non-null  float64
```

Dataset Two: Impute Remaining Missing Values

```
# fires_raw["DISCOVERY_DATE"]
# pd.to_datetime(fires_raw["DISCOVERY_DATE"]).dt.month.unique()
# fires_raw["DISCOVERY_DOY"].unique()

def doy_to_month(doy):
  if doy <= 31:
    return 1
  elif doy <= 59:
    return 2
  elif doy <= 90:
    return 3
  elif doy <= 120:
    return 4
  elif doy <= 151:
    return 5
  elif doy <= 181:
    return 6
  elif doy <= 212:
    return 7
  elif doy <= 243: # August
    return 8
  elif doy <= 273:
    return 9
  elif doy <= 304:
    return 10
  elif doy <= 334:
    return 11
  else:
    return 12
```

```
# We have two output variables of interest: FIRE_SIZE and STAT_CAUSE_CODE.
# We also have output variables of interest: CONT_TIME, CONT_DOY, DISCOVERY_TIME
# We will drop any rows without a value for these
# We already know it is in california
fires_raw = fires_raw.dropna(subset=["FIRE_SIZE", "STAT_CAUSE_CODE", "STATE", "CONT_TIME", "DISCOVERY_TIME"])

# We will drop fire name as you can't learn from it
# Drop local incident id as it isn't relevant either
if "FIRE_NAME" in fires_raw.columns:
  del fires_raw["FIRE_NAME"]

if "LOCAL_INCIDENT_ID" in fires_raw.columns:
  del fires_raw["LOCAL_INCIDENT_ID"]
```

```
# Look at remaining missing values.
count_missing_data(fires_raw)
fires_raw.isna().sum()

fires_raw.info()
```

```
  Missing Ratio
<class 'pandas.core.frame.DataFrame'>
Int64Index: 91184 entries, 0 to 1880460
Data columns (total 26 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   OBJECTID                    91184 non-null  int64  
 1   FOD_ID                      91184 non-null  int64  
 2   FPA_ID                      91184 non-null  object 
 3   SOURCE_SYSTEM_TYPE          91184 non-null  object 
 4   SOURCE_SYSTEM               91184 non-null  object 
 5   NWCG_REPORTING_AGENCY       91184 non-null  object 
 6   NWCG_REPORTING_UNIT_ID      91184 non-null  object 
 7   NWCG_REPORTING_UNIT_NAME    91184 non-null  object 
 8   SOURCE_REPORTING_UNIT       91184 non-null  object 
 9   SOURCE_REPORTING_UNIT_NAME  91184 non-null  object 
 10  FIRE_YEAR                   91184 non-null  int64  
 11  DISCOVERY_DATE              91184 non-null  float64
 12  DISCOVERY_DOY               91184 non-null  int64  
 13  DISCOVERY_TIME              91184 non-null  float64
 14  STAT_CAUSE_CODE             91184 non-null  float64
 15  STAT_CAUSE_DESCR            91184 non-null  object 
 16  CONT_DATE                   91184 non-null  float64
 17  CONT_DOY                    91184 non-null  float64
 18  CONT_TIME                   91184 non-null  float64
 19  FIRE_SIZE                   91184 non-null  float64
 20  FIRE_SIZE_CLASS             91184 non-null  object 
 21  LATITUDE                    91184 non-null  float64
 22  LONGITUDE                   91184 non-null  float64
 23  OWNER_CODE                  91184 non-null  float64
 24  OWNER_DESCR                 91184 non-null  object 
 25  STATE                       91184 non-null  object 
dtypes: float64(10), int64(4), object(12)
```

```
# Value imputation should be complete.
count_missing_data(fires_raw)
fires_raw.isna().sum()
```

```
  Missing Ratio
OBJECTID                      0
FOD_ID                        0
FPA_ID                        0
SOURCE_SYSTEM_TYPE            0
SOURCE_SYSTEM                 0
NWCG_REPORTING_AGENCY         0
NWCG_REPORTING_UNIT_ID        0
NWCG_REPORTING_UNIT_NAME      0
SOURCE_REPORTING_UNIT         0
SOURCE_REPORTING_UNIT_NAME    0
FIRE_YEAR                     0
DISCOVERY_DATE                0
DISCOVERY_DOY                 0
DISCOVERY_TIME                0
STAT_CAUSE_CODE               0
STAT_CAUSE_DESCR              0
CONT_DATE                     0
CONT_DOY                      0
CONT_TIME                     0
FIRE_SIZE                     0
FIRE_SIZE_CLASS               0
LATITUDE                      0
LONGITUDE                     0
OWNER_CODE                    0
OWNER_DESCR                   0
STATE                         0
```

Next, we will add the discovery and containment months.

```
fires_raw["DISCOVERY_MONTH"] = fires_raw["DISCOVERY_MONTH"] = fires_raw["DISCOVERY_DOY"].apply(doy_to_month)
fires_raw["CONTAINMENT_MONTH"] = fires_raw["CONT_DOY"].apply(doy_to_month)
```

```
fires_raw["DISCOVERY_MONTH"].unique()
```

```
array([ 2,  5,  6,  7,  3,  9, 10, 11,  4,  8,  1, 12])
```

Next, we split up columns into types by hand. We find this is the most precise way to do it.

```
display(fires_raw)

# SPECIAL COLS
FIRES_SPECIAL_COLS = ["FOD_ID", "LATITUDE", "LONGITUDE"]

# TIME RELATED COLS
FIRE_TIME_COLS = ["FIRE_YEAR", "DISCOVERY_DOY", "DISCOVERY_TIME", "CONT_DOY", "CONT_TIME", "DISCOVERY_MONTH", "CONTAINMENT_MONTH"]
FIRE_TIME_DATETIME = ["DISCOVERY_DATE", "CONT_DATE"]

# CATEGORICAL COLS
FIRES_CAT_COLS = ["SOURCE_SYSTEM_TYPE", "SOURCE_SYSTEM", "NWCG_REPORTING_AGENCY", "OWNER_DESCR", "NWCG_REPORTING_UNIT_NAME", "SOURCE_REPORTING_UNIT_NAME"]

# REDUNDANT COLS
FIRES_CAT_COL_REDUNDANT = ["SOURCE_REPORTING_UNIT", "OWNER_CODE", "NWCG_REPORTING_UNIT_ID"]

# EXTRA COLS TO DROP
FIRES_DROPS = ["OBJECTID", "FPA_ID"]

# OUTPUT COLS
OUTPUT_VALUES = ["FIRE_SIZE", "FIRE_SIZE_CLASS", "STAT_CAUSE_CODE", "STAT_CAUSE_DESCR", "CONT_DOY", "CONT_TIME"]
```

	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	FIRE_YEA
0	1	1	FS-1418826	FED	FS-FIRESTAT	FS	USCAPNF	Plumas National Forest	511	Plumas National Forest	200
1	2	2	FS-1418827	FED	FS-FIRESTAT	FS	USCAENF	Eldorado National Forest	503	Eldorado National Forest	200
2	3	3	FS-1418835	FED	FS-FIRESTAT	FS	USCAENF	Eldorado National Forest	503	Eldorado National Forest	200
3	4	4	FS-1418845	FED	FS-FIRESTAT	FS	USCAENF	Eldorado National Forest	503	Eldorado National Forest	200
4	5	5	FS-1418847	FED	FS-FIRESTAT	FS	USCAENF	Eldorado National Forest	503	Eldorado National Forest	200
...	...	...	...	...	...	...	...	...	...	...	
1880456	1880457	300348328	2015CAIRS27369138	NONFED	ST-CACDF	ST/C&L	USCATGU	Tehama-Glenn Unit	CATGU	Tehama-Glenn Unit	201
1880457	1880458	300348354	2015CAIRS28234594	NONFED	ST-CACDF	ST/C&L	USCASHU	Shasta-Trinity Unit	CASHU	Shasta-Trinity Unit	201
1880458	1880459	300348361	2015CAIRS27957490	NONFED	ST-CACDF	ST/C&L	USCAHUU	Humboldt-Del Norte Unit	CAHUU	Humboldt-Del Norte Unit	201
1880459	1880460	300348362	2015CAIRS28291374	NONFED	ST-CACDF	ST/C&L	USCALNU	Sonoma-Lake Napa Unit	CALNU	Sonoma-Lake Napa Unit	201
1880460	1880461	300348363	2015CAIRS29019636	NONFED	ST-CACDF	ST/C&L	USCASHU	Shasta-Trinity Unit	CASHU	Shasta-Trinity Unit	201

```
# Drop the last irrelevant columns.
if "OBJECTID" in fires_raw.columns:
  del fires_raw["OBJECTID"]
if "FPA_ID" in fires_raw.columns:
  del fires_raw["FPA_ID"]

# Drop redundant cols
for col in FIRES_CAT_COL_REDUNDANT:
  if col in fires_raw.columns:
    del fires_raw[col]
```

Docs:

Create a df where we have land temperature for every city in Cali using a dataset that maps lat/long to zipcodes and filters based on that

Create a model that uses knn to predict the acreage of a fire after it has started

Create a model that uses linear regression to predict the acreage of a fire after it has started

Create a model that uses svm to predict

Dataset we can use for city, state, latitude, longitude calculations https://github.com/kelvins/US-Cities-Database

Next, we will load in the third climate data set.

For each fire, we want to match it with weather data from the following time periods

1. Weather from month prior to fire occuring
2. Weather from second month from fire occuring
3. Weather from prior year of same month.

TBD How we will handle rows that don't have these matches....

```
import pandas as pd
noaa_weather = pd.read_csv("/content/drive/Shareddrives/Data Science 303 Group Project/csv/noaa_weather/noaa_CA_1992_2016_weather_2781174_CLEANED.csv")
display(noaa_weather)
```

	Unnamed: 0	STATION	LATITUDE	LONGITUDE	ELEVATION	STATION_NAME	YEAR	MONTH	NUM_COOLING_DEGREE_DAY_CUMULATIVE	NUM_COOLING_DEGREE_DAY	NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT	NUM_DAYS_WITH_MIN_T
0	0	USR0000CCOH	39.8717	-121.7689	-0.913217	COHASSET	1992	1	-1.239760	-0.886171	-0.155696	
1	1	USR0000CCOH	39.8717	-121.7689	-0.913217	COHASSET	1992	2	-0.850883	-0.338919	-0.155696	
2	2	USR0000CCOH	39.8717	-121.7689	-0.913217	COHASSET	1992	3	-0.850883	-0.886171	-0.155696	
3	3	USR0000CCOH	39.8717	-121.7689	-0.913217	COHASSET	1992	4	-0.316022	0.351139	-0.155696	
4	4	USR0000CCOH	39.8717	-121.7689	-0.913217	COHASSET	1992	5	0.477971	1.449139	-0.155696	
...	...	...	...	...	...	...	...	...	...	...	...	
5774	8394	USR0000CHNM	36.5625	-117.4736	1.197506	HUNTER_MOUNTAIN	2011	11	0.739154	-0.886171	-0.155696	
5775	8199	USR0000CHNM	36.5625	-117.4736	1.197506	HUNTER_MOUNTAIN	1994	12	0.728818	-0.886171	-0.155696	
5776	8386	USR0000CHNM	36.5625	-117.4736	1.197506	HUNTER_MOUNTAIN	2010	12	0.772949	-0.886171	-0.155696	
5777	8386	USR0000CHNM	36.5625	-117.4736	1.197506	HUNTER_MOUNTAIN	2011	12	0.772949	-0.886171	-0.155696	
5778	8386	USR0000CHNM	36.5625	-117.4736	1.197506	HUNTER_MOUNTAIN	2012	12	0.772949	-0.886171	-0.155696	

5779 rows × 21 columns

Here, we add a column indicating the distance of the fire from each station. Be patient, takes about 5 minutes to run!

```
# STEP 1: Pair every fire in the database with it's closest station
station_lat_long_pairs = noaa_weather[["STATION_NAME", "LATITUDE", "LONGITUDE"]]
station_lat_long_pairs = station_lat_long_pairs.drop_duplicates()
# station_lat_long_pairs["DISTANCE_FROM_STATION_NAME"] = station_lat_long_pairs["STATION_NAME"].apply(lambda name: ("DISTANCE_FROM_" + name))
station_lat_long_pairs.set_index("STATION_NAME", inplace=True)

# display(station_lat_long_pairs)
# display(station_lat_long_pairs)

fire_id_lat_long_pairs = fires_raw[["FOD_ID", "LATITUDE", "LONGITUDE"]]

def haversine_distance_to_each_station(fire_data):
  fire_lat = fire_data["LATITUDE"]
  fire_long = fire_data["LONGITUDE"]

  distances = station_lat_long_pairs.apply(lambda station_data: haversine((fire_lat, fire_long), (station_data["LATITUDE"], station_data["LONGITUDE"])), axis=1, result_type="expand")
  fire_data = fire_data.append(distances)
  return fire_data

fires_raw = fires_raw.apply(haversine_distance_to_each_station, axis=1)
```

```
display(fires_raw)
fires_raw.to_csv("/content/drive/Shareddrives/Data Science 303 Group Project/csv/cleaned_fires_data_with_distance_to_each_station_2.csv")
```

	FOD_ID	SOURCE_SYSTEM_TYPE	SOURCE_SYSTEM	NWCG_REPORTING_AGENCY	NWCG_REPORTING_UNIT_NAME	SOURCE_REPORTING_UNIT_NAME	FIRE_YEAR	DISCOVERY_DATE	DISCOVERY_DOY	DISCOVERY_TIME	STAT_CAUSE_CODE	STAT_CAU
0	1	FED	FS-FIRESTAT	FS	Plumas National Forest	Plumas National Forest	2005	2453403.5	33	1300.0	9.0	Mis
1	2	FED	FS-FIRESTAT	FS	Eldorado National Forest	Eldorado National Forest	2004	2453137.5	133	845.0	1.0	
2	3	FED	FS-FIRESTAT	FS	Eldorado National Forest	Eldorado National Forest	2004	2453156.5	152	1921.0	5.0	Deb
3	4	FED	FS-FIRESTAT	FS	Eldorado National Forest	Eldorado National Forest	2004	2453184.5	180	1600.0	1.0	
4	5	FED	FS-FIRESTAT	FS	Eldorado National Forest	Eldorado National Forest	2004	2453184.5	180	1600.0	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	
1880456	300348328	NONFED	ST-CACDF	ST/C&L	Tehama-Glenn Unit	Tehama-Glenn Unit	2015	2457187.5	165	1714.0	13.0	Missing/
1880457	300348354	NONFED	ST-CACDF	ST/C&L	Shasta-Trinity Unit	Shasta-Trinity Unit	2015	2457295.5	273	2357.0	7.0	
1880458	300348361	NONFED	ST-CACDF	ST/C&L	Humboldt-Del Norte Unit	Humboldt-Del Norte Unit	2015	2457235.5	213	1331.0	1.0	
1880459	300348362	NONFED	ST-CACDF	ST/C&L	Sonoma-Lake Napa Unit	Sonoma-Lake Napa Unit	2015	2457170.5	148	1420.0	9.0	Mis
1880460	300348363	NONFED	ST-CACDF	ST/C&L	Shasta-Trinity Unit	Shasta-Trinity Unit	2015	2457291.5	269	1726.0	13.0	Missing/

Nice, now that we have distances, we will make a quick model / sanity check for this data.

Eventually, we want to include all the details of the nearest stations, but we might be able to learn on distance alone right now.

Here we reload our file to avoid expensive computation

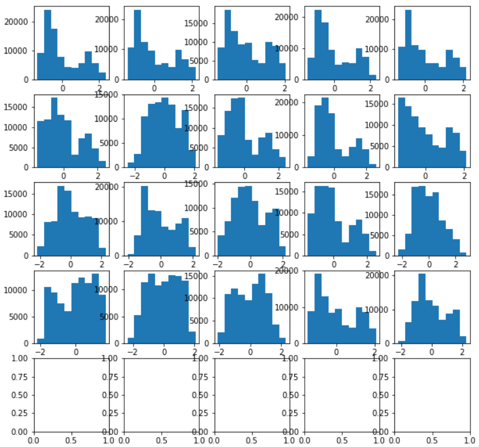
```
fires_with_distance = pd.read_csv("/content/drive/Shareddrives/Data Science 303 Group Project/csv/cleaned_fires_data_with_distance_to_each_station_2.csv")
```

Plot the normalized distances from each fire to each station

```
import math
# Normalize distance to the stations
num_stations = len(station_lat_long_pairs.index)
nr = nc = math.ceil(num_stations ** .5)
fig, ax = plt.subplots(nr, nc, figsize=(10,10))

```

```
count = 0
for station in station_lat_long_pairs.index:
  row = count // nr
  col = count % nc
  fires_with_distance[station] = (fires_with_distance[station] - fires_with_distance[station].mean()) / fires_with_distance[station].std()
  ax[row][col].hist(fires_with_distance[station])
  count += 1
```



Next, we iterate over every fire and pair it with its prior month's weather

```
display(noaa_weather["MONTH"].unique())
display(fires_with_distance["DISCOVERY_MONTH"].unique())

# display(fires_with_distance)
# print(fires_with_distance.info())
# Get the closest station for each fire.
station_names = station_lat_long_pairs.index

REDUNDANT_LABELS = ["STATION", "STATION_NAME", "YEAR", "MONTH", "Unnamed: 0"]

NUM_MONTHS_PRIOR = 16 # Given the closest station, how many months back do we want to record in our vector?
NUM_CLOSEST_STATIONS = 0 # How many of the closest stations do we want to consider?

#get_matching_station = lambda weather_table, station, month, year: (weather_table[((weather_table["STATION_NAME"] == station) & (weather_table["MONTH"] == month) & (weather_table["YEAR"] == year))])

def get_matching_station(weather_table, station, month, year):
  return (weather_table[((weather_table["STATION_NAME"] == station) & (weather_table["MONTH"] == month) & (weather_table["YEAR"] == year))])

def prior_month_and_year(month, n, year):
  """
  Given a month and year pair, return the month and year pair for n months prior
  """
  p_month = ((month - n - 1) % 12) + 1
  p_year = year
  if p_month > month: # Indicates that the prior month occured in the prior year
    p_year += -1
  return p_month, p_year


# Make sure this function is working correctly
assert(prior_month_and_year(5, 3, 2012) == (2, 2012))
assert(prior_month_and_year(5, 5, 2012) == (12, 2011))
assert(prior_month_and_year(1, 1, 2012) == (12, 2011))
assert(prior_month_and_year(2, 7, 2012) == (7, 2011))

def find_n_closest_stations(fire_row):
  # Sort the values:
  # We can do this faster than O(n log(n)), but it isn't really worth it.
  # print(fire_row)
  distance = fire_row[station_names].sort_values()

  fire_month = fire_row["DISCOVERY_MONTH"]
  fire_year = fire_row["FIRE_YEAR"]
  primary_station = distance.index[0]

  # GET WEATHER FROM THE ONE CLOSEST STATION, FOR THE NUM_MONTHS_PRIOR MONTHS
  for j in range(1, NUM_MONTHS_PRIOR + 1):
    p_month, p_year = prior_month_and_year(fire_month, j, fire_year)
    prefix = f"PRIMARY_STATION_{j}_MONTHS_PRIOR_"
    weather_j_months_prior = get_matching_station(noaa_weather, primary_station, p_month, p_year)

    if not weather_j_months_prior.empty:
      weather_j_months_prior = weather_j_months_prior.iloc[0]
      weather_j_months_prior = weather_j_months_prior.drop(REDUNDANT_LABELS)
      weather_j_months_prior = weather_j_months_prior.add_prefix(prefix)
      fire_row = fire_row.append(weather_j_months_prior)
    else:
      print(f"Failed to find weather for j={j} months prior. Searched for {primary_station}:{p_month}:{p_year}")
      break
      # print(weather_j_months_prior)
      # assert(False)

  # GET WEATHER FOR NUM_CLOSEST_STATIONS STATIONS
  p_month, p_year = prior_month_and_year(fire_month, 1, fire_year)
  for i in range(1, NUM_CLOSEST_STATIONS + 1):
    station_name = distance.index[i]
    closest_station_prefix = f"CLOSEST_STATION_{i + 1}_"
    fire_row[closest_station_prefix] = station_name

    # Get the weather for that month, year, and station
    # weather = noaa_weather[((noaa_weather["STATION_NAME"] == station_name) & (noaa_weather["MONTH"] == month) & (noaa_weather["YEAR"] == year))]
    weather = get_matching_station(noaa_weather, station_name, p_month, p_year)
    if not weather.empty:
      weather = weather.iloc[0]
      weather = weather.drop(REDUNDANT_LABELS)
      weather = weather.add_prefix(closest_station_prefix)
      fire_row = fire_row.append(weather)
    else:
      print(f"Failed to find closest station match for {station_name}:{p_month}:{p_year}")
      break
      # print(weather)
      # print(get_matching_station(noaa_weather, station_name, p_month, p_year))
      # assert(False)

  return fire_row


fires_with_full_weather = fires_with_distance.apply(find_n_closest_stations, axis=1)
fires_with_full_weather = fires_with_full_weather.dropna() # Drop any records that we couldn't find past matches for.
# fires_with_full_weather = fires_raw[:5].apply(find_n_closest_stations, axis=1)
display(fires_with_full_weather)


```

```
get_matching_station(noaa_weather, "HELL_HOLE", 4, 2004)
```

	Unnamed: 0	STATION	LATITUDE	LONGITUDE	ELEVATION	STATION_NAME	YEAR	MONTH	NUM_COOLING_DEGREE_DAY_CUMULATIVE	NUM_COOLING_DEGREE_DAY	NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT	NUM_DAYS_WITH_MIN_TEMP_B
5456	2316	USR0000CHEL	39.0717	-120.4217	0.525056	HELL_HOLE	2004	4	-1.23976	-0.886171	-0.155696	

```
fires_with_full_weather.to_csv("/content/drive/Shareddrives/Data Science 303 Group Project/csv/cleaned_fires_data_with_four_closest_stations_nov21_dont_use_this.csv")
```

```
import seaborn as sns
corr = fires_with_full_weather.iloc[:10000].select_dtypes(include = "number").corr()

plt.figure(figsize=(10, 10))
sns.heatmap(corr,
cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1, square=True);
```

More cleaning to do!

```
# We want to get a list of all the columns that are numerical and havent been normalized yet
numeric_features = fires_with_full_weather.select_dtypes(include="number")
categorical_features = fires_with_full_weather.select_dtypes(include="object")
print(numeric_features.info(verbose=True))
print(categorical_features.info(verbose=True))
```