DengAI - Predicting Disease Spread

Main objective of the analysis that specifies whether your model will be focused on prediction or interpretation

The project is focus solely on prediction of dengue cases.

Brief description of the data set and a summary of its attributes

The data for this competition comes from multiple sources aimed at supporting the Predict the Next Pandemic Initiative. Dengue surveillance data is provided by the U.S. Centers for Disease Control and prevention, as well as the Department of Defense's Naval Medical Research Unit 6 and the Armed Forces Health Surveillance Center, in collaboration with the Peruvian government and U.S. universities. Environmental and climate data is provided by the National Oceanic and Atmospheric Administration (NOAA), an agency of the U.S. Department of Commerce.

Problem description

Your goal is to predict the total_cases label for (San Juan, year, weekofyear) in the test set. There is this city, San Juan, with test data for it spanning 5 and 3 years respectively. The test set is a pure future hold-out, meaning the test data are sequential and non-overlapping with any of the training data. Throughout, missing values have been filled as NaNs.

The features in this dataset

You are provided the following set of information on a (year, weekofyear) timescale:

Field	Description
city	City abbreviations: sj for San Juan
week_start_date	Date given in yyyy-mm-dd format
station_max_temp_c	Maximum temperature
station_min_temp_c	Minimum temperature
station_avg_temp_c	Average temperature
station_precip_mm	Total precipitation
station_diur_temp_rng_c	Diurnal temperature range
precipitation_amt_mm	Total precipitation
reanalysis_sat_precip_amt_mm	Total precipitation
reanalysis_dew_point_temp_k	Mean dew point temperature
reanalysis_air_temp_k	Mean air temperature
reanalysis_relative_humidity_percent	Mean relative humidity
reanalysis_specific_humidity_g_per_kg	Mean specific humidity
reanalysis_precip_amt_kg_per_m2	Total precipitation
reanalysis_max_air_temp_k	Maximum air temperature
reanalysis_min_air_temp_k	Minimum air temperature
reanalysis_avg_temp_k	Average air temperature
reanalysis_tdtr_k	Diurnal temperature range
ndvi_se	Pixel southeast of city centroid
ndvi_sw	Pixel southwest of city centroid
ndvi_ne	Pixel northeast of city centroid
ndvi_nw	Pixel northwest of city centroid

Brief summary of data exploration and actions taken for data cleaning and feature engineering

Data Exploration includes data summary, statistics, relevant graphs to find any relationships within.

As for data cleaning, we will check for missing values and decide what imputation method. We also check for data duplicates and outliers. Finally perform binary encoding before model training.

Import Libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn

from sklearn.linear_model import LinearRegression, Ridge, Lasso

from sklearn.model_selection import cross_val_score, train_test_split, GridSearchC
V, RandomizedSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler, OneHo
tEncoder, PolynomialFeatures
from sklearn.metrics import mean_absolute_error, mean_squared_error,r2_score


%matplotlib inline
sns.set_style('dark')
sns.set(font_scale=1.2)

import warnings
warnings.filterwarnings('ignore')
from pycaret.regression import *

np.random.seed(123)

pd.options.display.max_columns= None
#pd.options.display.max_rows = None
```

In [2]:

```
df = pd.read_csv("dengue_features_train.csv")
```

In [3]:

```
df
```

Out[3]:

	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm
0	sj	1990	18	30/4/1990	0.122600	0.103725	0.198483	0.177617	12.42
1	sj	1990	19	7/5/1990	0.169900	0.142175	0.162357	0.155486	22.82
2	sj	1990	20	14/5/1990	0.032250	0.172967	0.157200	0.170843	34.54
3	sj	1990	21	21/5/1990	0.128633	0.245067	0.227557	0.235886	15.36
4	sj	1990	22	28/5/1990	0.196200	0.262200	0.251200	0.247340	7.52
...	...	...	...	...	...	...	...	...	...
931	sj	2008	13	25/3/2008	0.077850	-0.039900	0.310471	0.296243	27.19
932	sj	2008	14	1/4/2008	-0.038000	-0.016833	0.119371	0.066386	3.82
933	sj	2008	15	8/4/2008	-0.155200	-0.052750	0.137757	0.141214	16.96
934	sj	2008	16	15/4/2008	0.001800	NaN	0.203900	0.209843	0.00
935	sj	2008	17	22/4/2008	-0.037000	-0.010367	0.077314	0.090586	0.00

936 rows × 25 columns

Dataset has 2 categorical features and 23 numeric features.

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 936 entries, 0 to 935
Data columns (total 25 columns):
 #   Column                                 Non-Null Count  Dtype  
---  ------                                 --------------  -----  
 0   city                                   936 non-null    object 
 1   year                                   936 non-null    int64  
 2   weekofyear                             936 non-null    int64  
 3   week_start_date                        936 non-null    object 
 4   ndvi_ne                                745 non-null    float64
 5   ndvi_nw                                887 non-null    float64
 6   ndvi_se                                917 non-null    float64
 7   ndvi_sw                                917 non-null    float64
 8   precipitation_amt_mm                   927 non-null    float64
 9   reanalysis_air_temp_k                  930 non-null    float64
 10  reanalysis_avg_temp_k                  930 non-null    float64
 11  reanalysis_dew_point_temp_k            930 non-null    float64
 12  reanalysis_max_air_temp_k              930 non-null    float64
 13  reanalysis_min_air_temp_k              930 non-null    float64
 14  reanalysis_precip_amt_kg_per_m2        930 non-null    float64
 15  reanalysis_relative_humidity_percent   930 non-null    float64
 16  reanalysis_sat_precip_amt_mm           927 non-null    float64
 17  reanalysis_specific_humidity_g_per_kg  930 non-null    float64
 18  reanalysis_tdtr_k                      930 non-null    float64
 19  station_avg_temp_c                     930 non-null    float64
 20  station_diur_temp_rng_c                930 non-null    float64
 21  station_max_temp_c                     930 non-null    float64
 22  station_min_temp_c                     930 non-null    float64
 23  station_precip_mm                      930 non-null    float64
 24  total_cases                            936 non-null    int64  
dtypes: float64(20), int64(3), object(2)
memory usage: 182.9+ KB
```

Summary of statistics below:

```
In [5]: df.describe(include='all').T
```

Out[5]:

	count	unique	top	freq	mean	std	min	2
city	936	1	sj	936	NaN	NaN	NaN	N
year	936	NaN	NaN	NaN	1998.83	5.21208	1990	19
weekofyear	936	NaN	NaN	NaN	26.5032	15.0219	1	13
week_start_date	936	936	24/9/1994	1	NaN	NaN	NaN	N
ndvi_ne	745	NaN	NaN	NaN	0.0579247	0.107153	-0.40625	0.00
ndvi_nw	887	NaN	NaN	NaN	0.0674691	0.0924788	-0.4561	0.0164
ndvi_se	917	NaN	NaN	NaN	0.177655	0.0571663	-0.0155333	0.1392
ndvi_sw	917	NaN	NaN	NaN	0.165956	0.0560733	-0.0634571	0.1291
precipitation_amt_mm	927	NaN	NaN	NaN	35.4708	44.6061	0	
reanalysis_air_temp_k	930	NaN	NaN	NaN	299.164	1.23643	295.939	298.1
reanalysis_avg_temp_k	930	NaN	NaN	NaN	299.277	1.21864	296.114	29
reanalysis_dew_point_temp_k	930	NaN	NaN	NaN	295.11	1.56994	289.643	293.8
reanalysis_max_air_temp_k	930	NaN	NaN	NaN	301.399	1.25893	297.8	30
reanalysis_min_air_temp_k	930	NaN	NaN	NaN	297.302	1.29471	292.6	29
reanalysis_precip_amt_kg_per_m2	930	NaN	NaN	NaN	30.4654	35.6281	0	10.8
reanalysis_relative_humidity_percent	930	NaN	NaN	NaN	78.5682	3.38949	66.7357	76.24
reanalysis_sat_precip_amt_mm	927	NaN	NaN	NaN	35.4708	44.6061	0	
reanalysis_specific_humidity_g_per_kg	930	NaN	NaN	NaN	16.5524	1.56092	11.7157	15.23
reanalysis_tdtr_k	930	NaN	NaN	NaN	2.51627	0.498892	1.35714	2.157
station_avg_temp_c	930	NaN	NaN	NaN	27.0065	1.41547	22.8429	25.84
station_diur_temp_rng_c	930	NaN	NaN	NaN	6.75737	0.835993	4.52857	
station_max_temp_c	930	NaN	NaN	NaN	31.608	1.7173	26.7	3
station_min_temp_c	930	NaN	NaN	NaN	22.6006	1.50628	17.8	2
station_precip_mm	930	NaN	NaN	NaN	26.7855	29.3258	0	6.8
total_cases	936	NaN	NaN	NaN	34.1806	51.3814	0	

Shape of dataset:

```
In [6]: df.shape
```

```
Out[6]: (936, 25)
```

```
In [7]: df.columns
```

```
Out[7]: Index(['city', 'year', 'weekofyear', 'week_start_date', 'ndvi_ne', 'ndvi_nw',
               'ndvi_se', 'ndvi_sw', 'precipitation_amt_mm', 'reanalysis_air_temp_k',
               'reanalysis_avg_temp_k', 'reanalysis_dew_point_temp_k',
               'reanalysis_max_air_temp_k', 'reanalysis_min_air_temp_k',
               'reanalysis_precip_amt_kg_per_m2',
               'reanalysis_relative_humidity_percent', 'reanalysis_sat_precip_amt_mm',
               'reanalysis_specific_humidity_g_per_kg', 'reanalysis_tdtr_k',
               'station_avg_temp_c', 'station_diur_temp_rng_c', 'station_max_temp_c',
               'station_min_temp_c', 'station_precip_mm', 'total_cases'],
              dtype='object')
```

Data Exploration

Dataset starts from 1990 to 2008

```
In [8]: df['year'].unique()
```

```
Out[8]: array([1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000,
               2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008], dtype=int64)
```

Weekofyear is mostly consistent figures except the last one

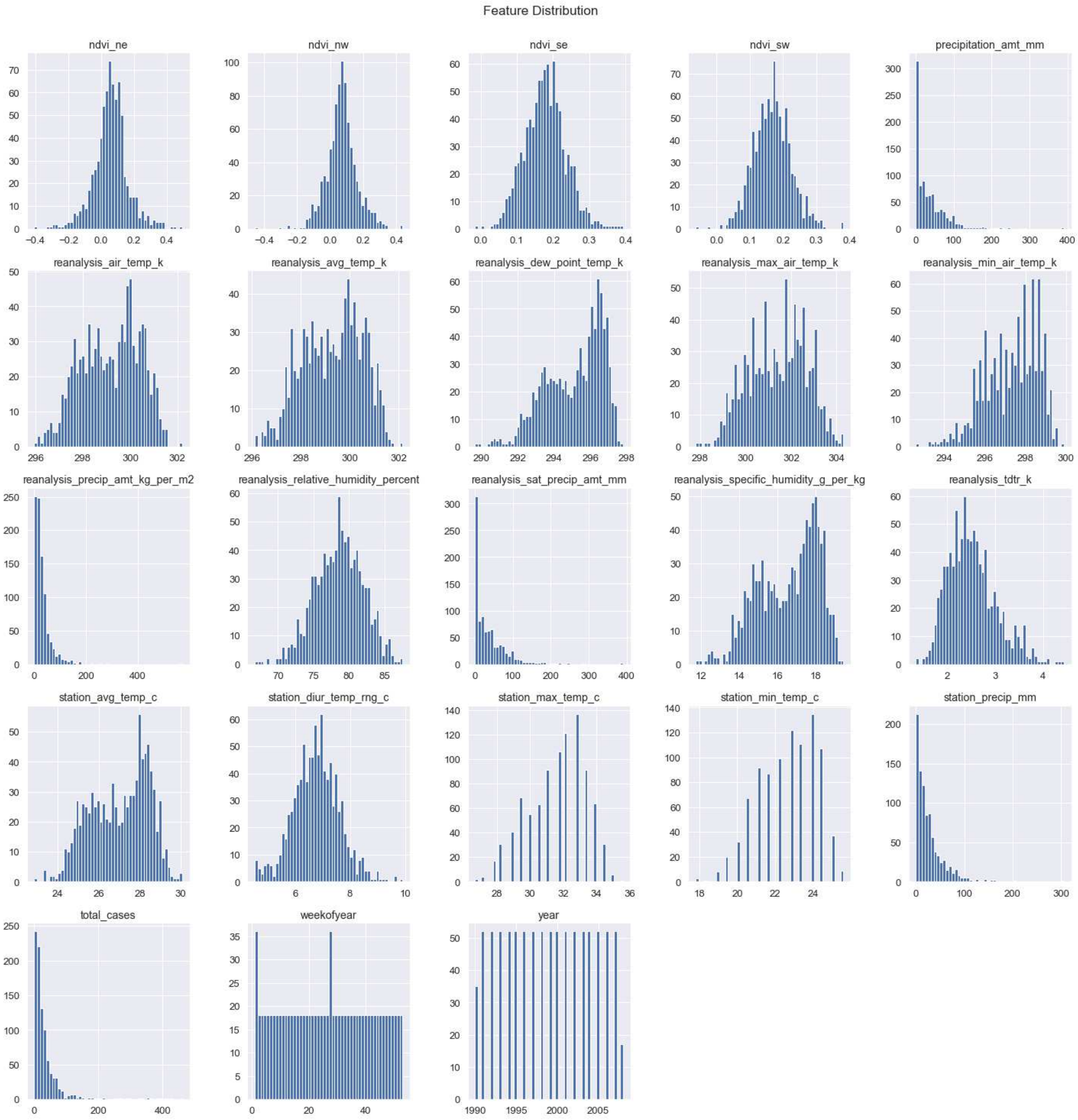
In [9]:
```
df['weekofyear'].value_counts()
```

Out[9]:
```
27    18
26    18
24    18
23    18
22    18
21    18
20    18
19    18
18    18
17    18
16    18
15    18
14    18
13    18
12    18
11    18
10    18
9     18
8     18
7     18
6     18
5     18
4     18
3     18
2     18
25    18
1     18
40    18
51    18
49    18
48    18
47    18
46    18
45    18
44    18
43    18
42    18
41    18
28    18
39    18
38    18
37    18
36    18
35    18
34    18
33    18
32    18
31    18
30    18
29    18
50    18
52    15
53     3
Name: weekofyear, dtype: int64
```

Data Visualization

The dataset is Normally distributed.

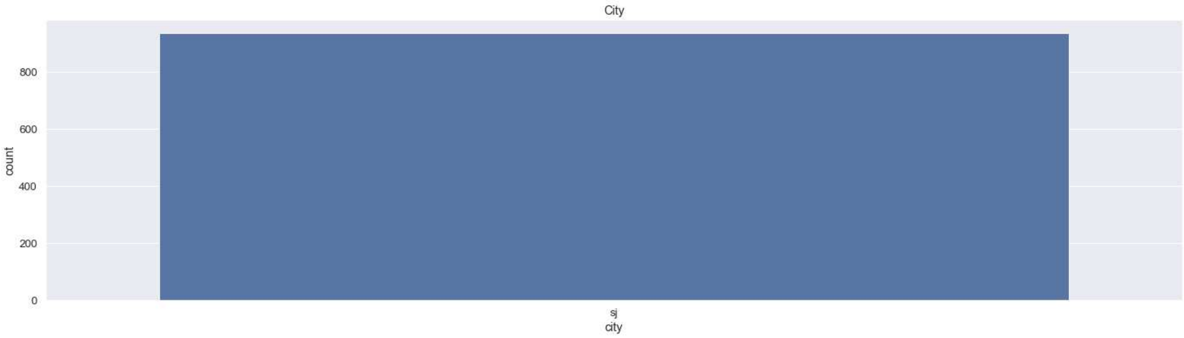
```
In [10]: df.hist(bins=50, figsize=(20,20))

         plt.suptitle('Feature Distribution', x=0.5, y=1.02, ha='center', fontsize='large')

         plt.tight_layout()

         plt.show();
```

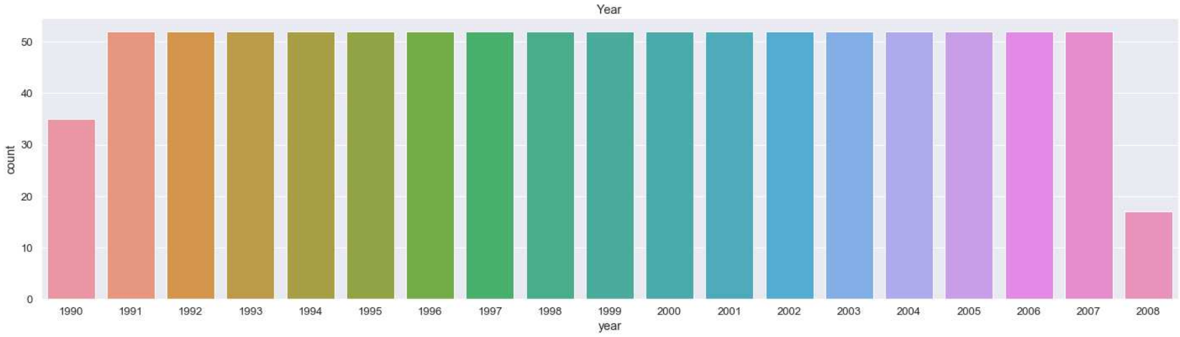


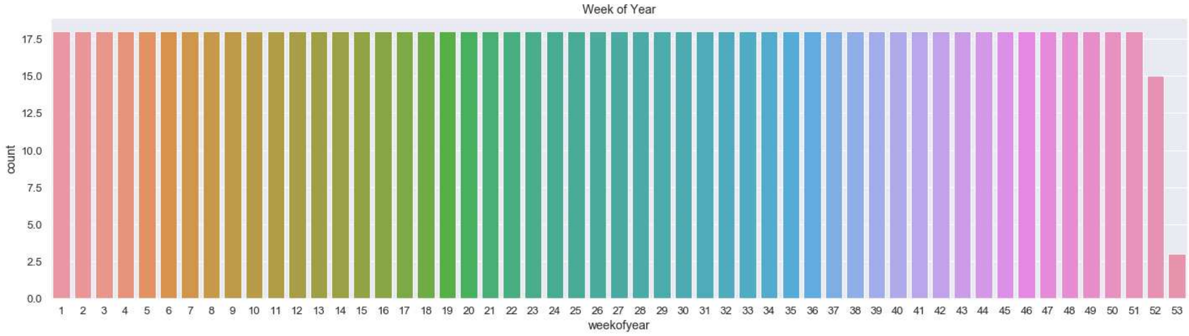
Below are each visuals of the data:

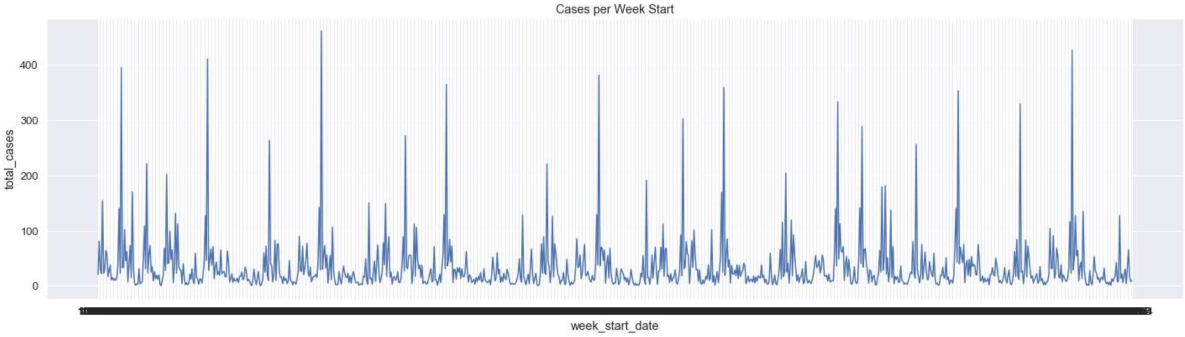
In [13]:

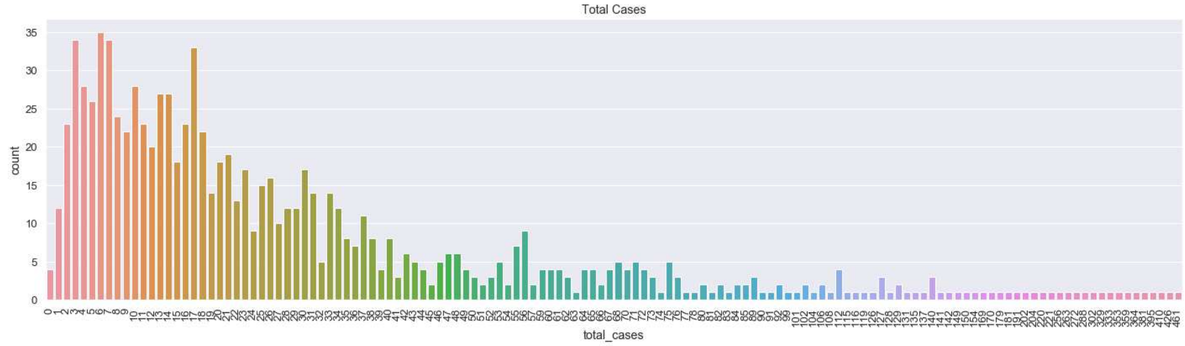
```
fig = plt.figure(figsize=(20,40))

plt.subplot(7,1,1)
plt.title("City")
sns.countplot(df.city)

plt.subplot(7,1,2)
plt.title("Year")
sns.countplot(df.year)

plt.subplot(7,1,3)
plt.title("Week of Year")
sns.countplot(df.weekofyear)

plt.subplot(7,1,4)
plt.title("Cases per Week Start")
sns.lineplot(x=df.week_start_date, y=df.total_cases)

plt.subplot(7,1,5)
plt.title("Total Cases")
plt.xticks(rotation=90)
sns.countplot(df.total_cases)

plt.subplot(7,1,6)
plt.title("Cases per Year")
sns.barplot(x=df.year,y=df.total_cases,data=df,ci=None)


plt.tight_layout()
plt.show()
```

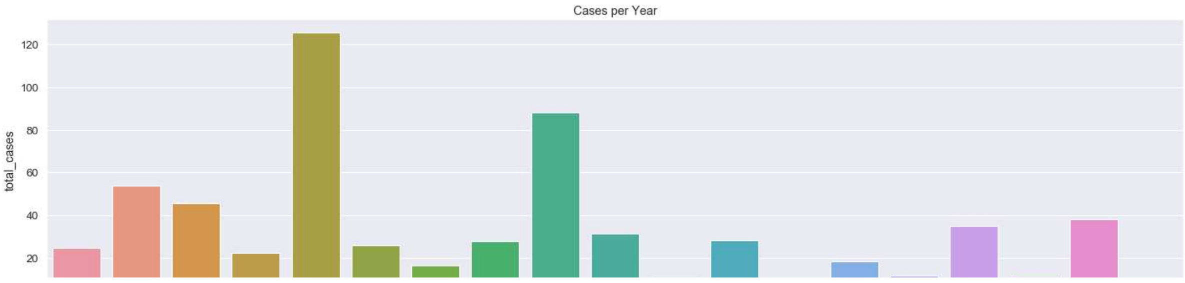

City
count
0
200
400
600
800
sj
city


Year
count
0
10
20
30
40
50
1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008
year


Week of Year
count
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
weekofyear

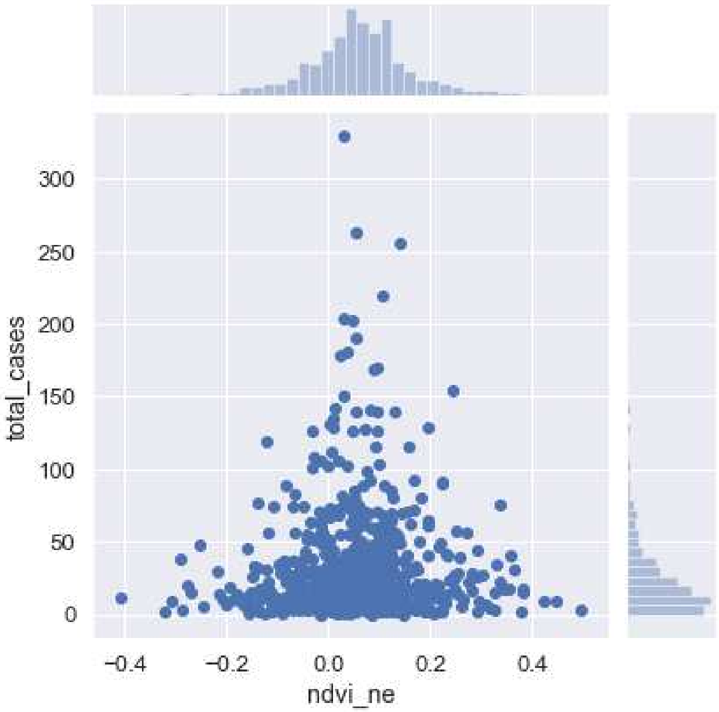

Cases per Week Start
total_cases
0
100
200
300
400
week_start_date

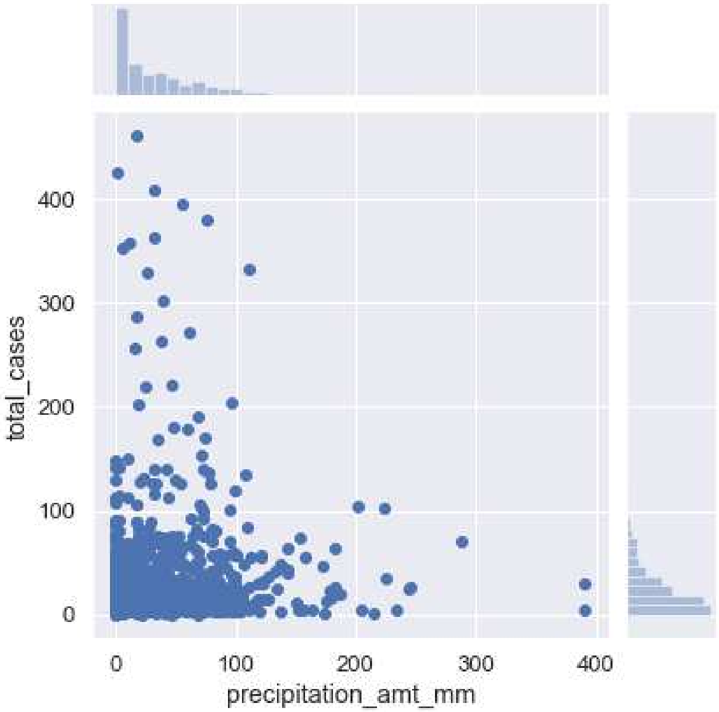

Total Cases
count
0
5
10
15
20
25
30
35
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 59 60 61 62 63 64 65 66 67 68 70 71 72 73 74 75 76 77 78 80 81 82 83 84 85 89 90 91 92 99 101 102 104 106 108 112 115 116 119 126 127 128 129 131 135 137 140 141 142 149 150 154 169 170 179 181 191 202 204 220 221 256 263 272 282 288 302 329 333 353 359 364 381 395 410 426 461
total_cases

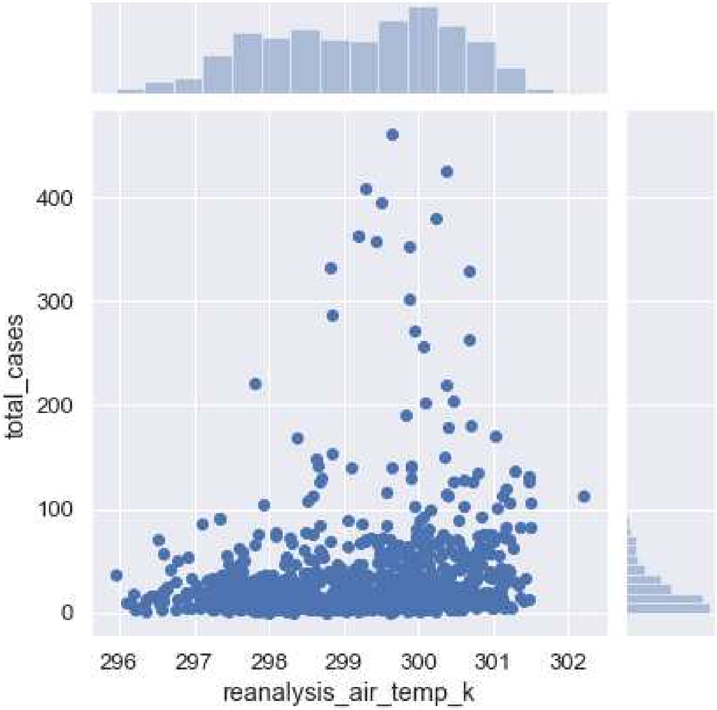

Cases per Year
total_cases
20
40
60
80
100
120

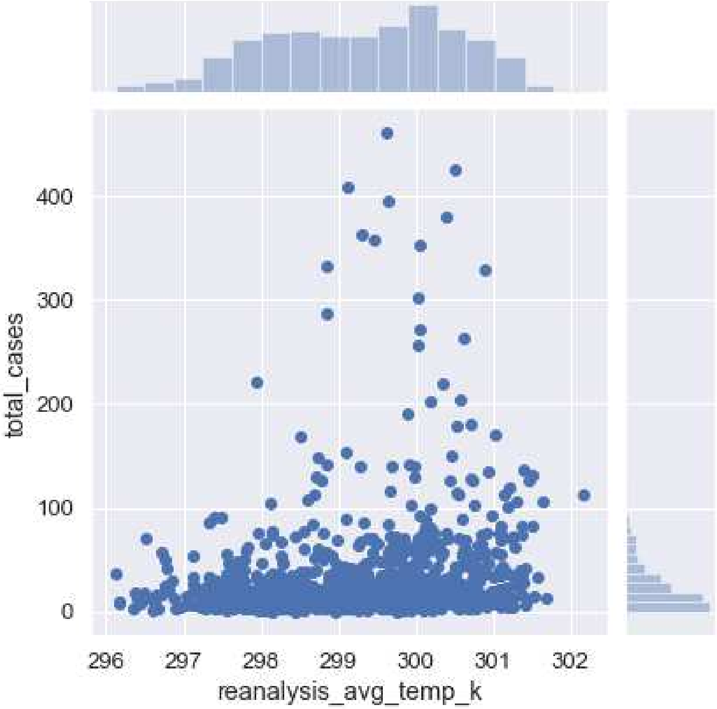
In [14]:

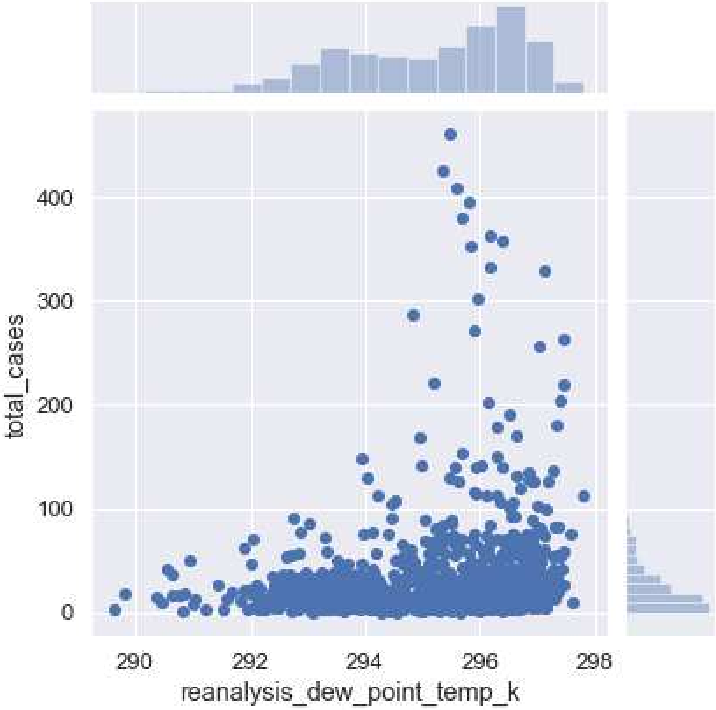
```
sns.jointplot(x='ndvi_ne', y='total_cases',data=df, kind='scatter')

sns.jointplot(x='precipitation_amt_mm', y='total_cases',data=df, kind='scatter')

sns.jointplot(x='reanalysis_air_temp_k', y='total_cases',data=df, kind='scatter')

sns.jointplot(x='reanalysis_avg_temp_k', y='total_cases',data=df, kind='scatter')

sns.jointplot(x='reanalysis_dew_point_temp_k', y='total_cases',data=df, kind='scatter')

sns.jointplot(x='reanalysis_precip_amt_kg_per_m2', y='total_cases',data=df, kind='scatter')

sns.jointplot(x='reanalysis_tdtr_k', y='total_cases',data=df, kind='scatter')

sns.jointplot(x='station_avg_temp_c', y='total_cases',data=df, kind='scatter')

sns.jointplot(x='station_diur_temp_rng_c', y='total_cases',data=df, kind='scatter')

sns.jointplot(x='station_precip_mm', y='total_cases',data=df, kind='scatter')


plt.show()
```

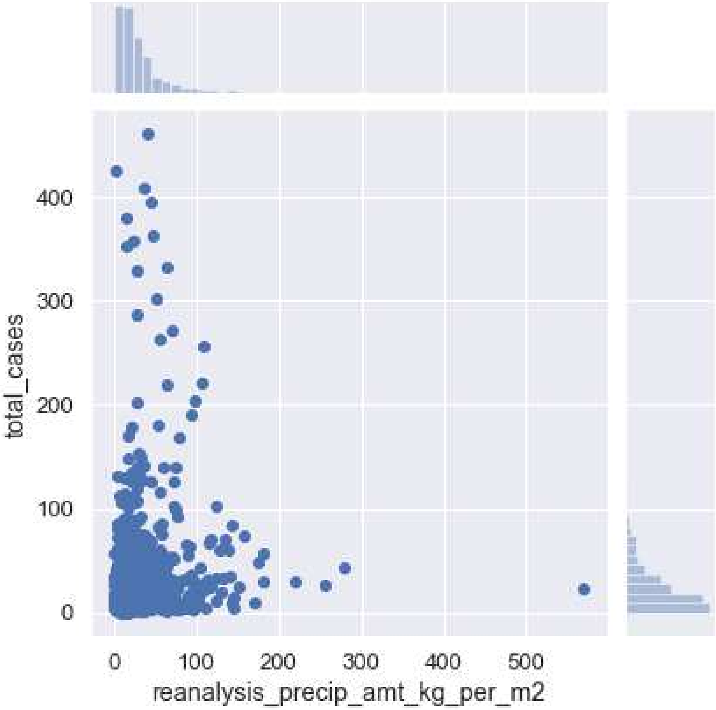

total_cases
300
250
200
150
100
50
0
−0.4
−0.2
0.0
0.2
0.4
ndvi_ne

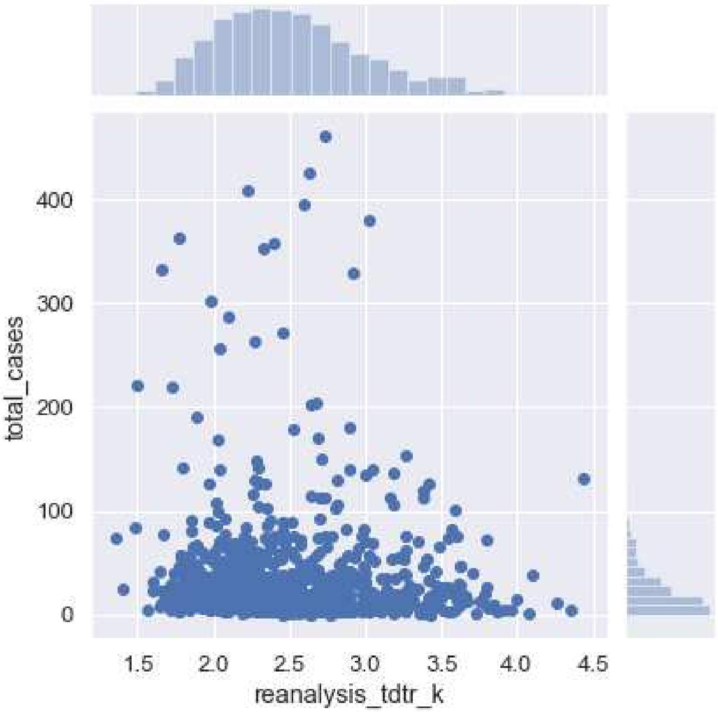

total_cases
400
300
200
100
0
0
100
200
300
400
precipitation_amt_mm

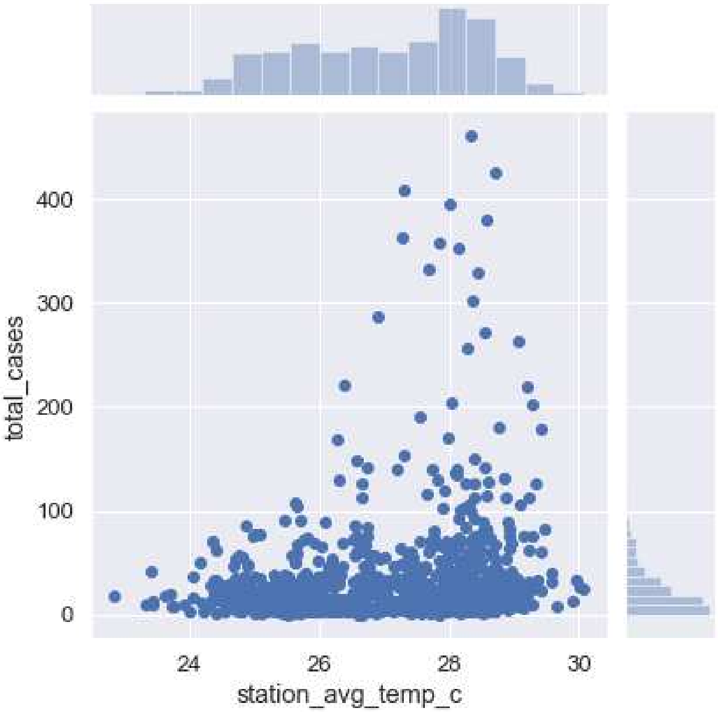

total_cases
0
100
200
300
400
296
297
298
299
300
301
302
reanalysis_air_temp_k

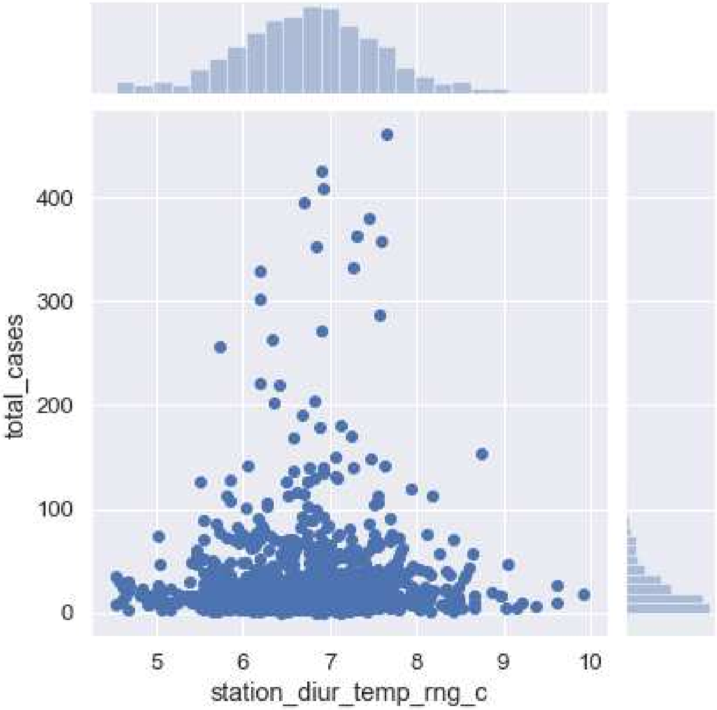

total_cases
0
100
200
300
400
296
297
298
299
300
301
302
reanalysis_avg_temp_k

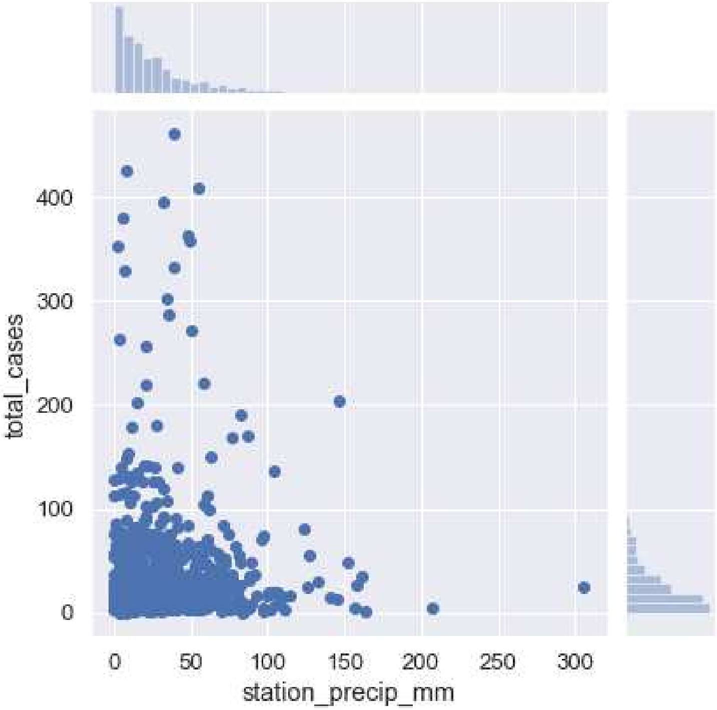

total_cases
0
100
200
300
400
290
292
294
296
298
reanalysis_dew_point_temp_k


total_cases
0
100
200
300
400
0
100
200
300
400
500
reanalysis_precip_amt_kg_per_m2


total_cases
400
300
200
100
0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
reanalysis_tdtr_k

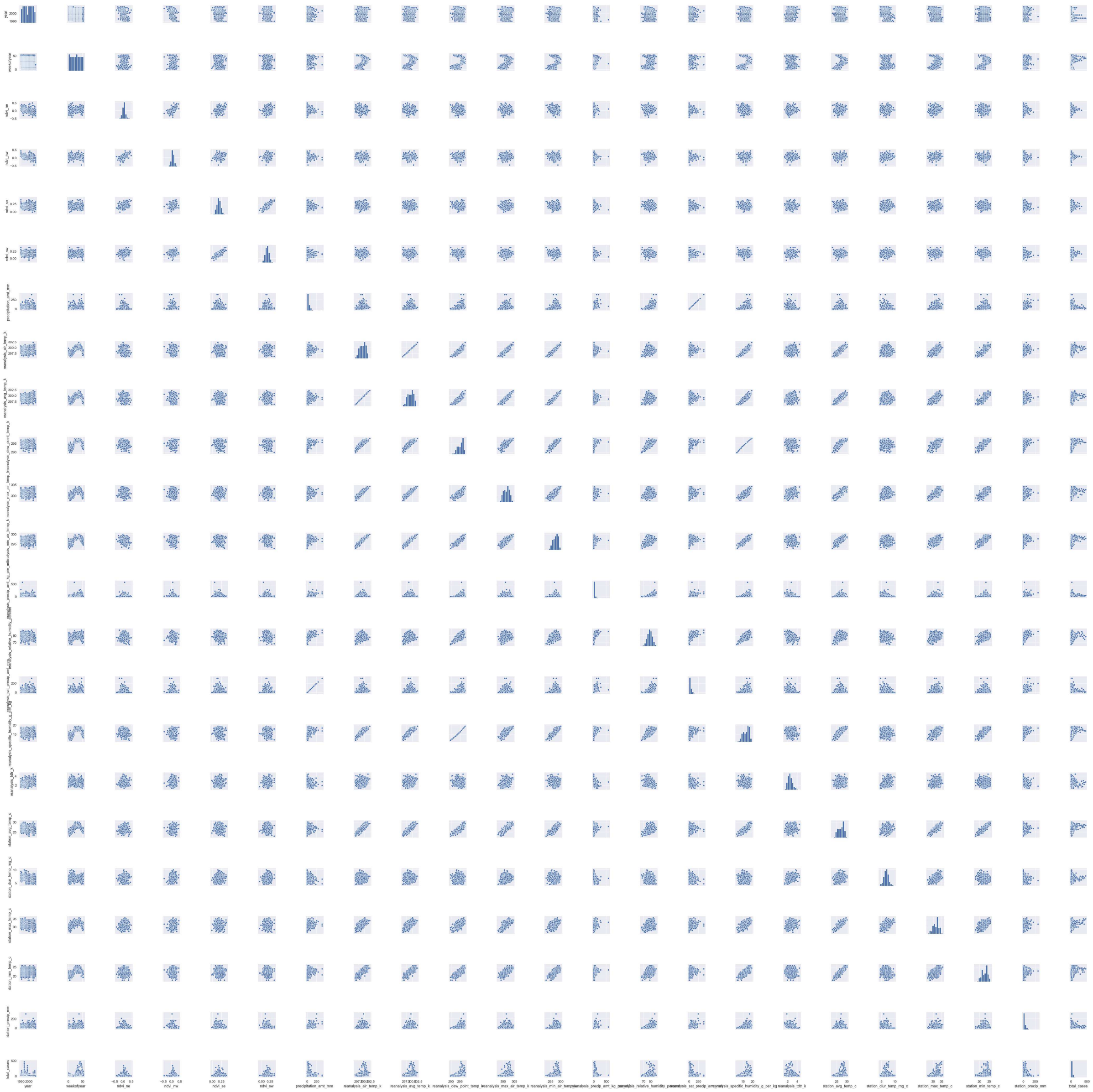

total_cases
400
300
200
100
0
24
26
28
30
station_avg_temp_c


total_cases
0
100
200
300
400
5
6
7
8
9
10
station_diur_temp_rng_c


total_cases
0
100
200
300
400
0
50
100
150
200
250
300
station_precip_mm

In [15]:

```
sns.pairplot(df)
plt.show()
```



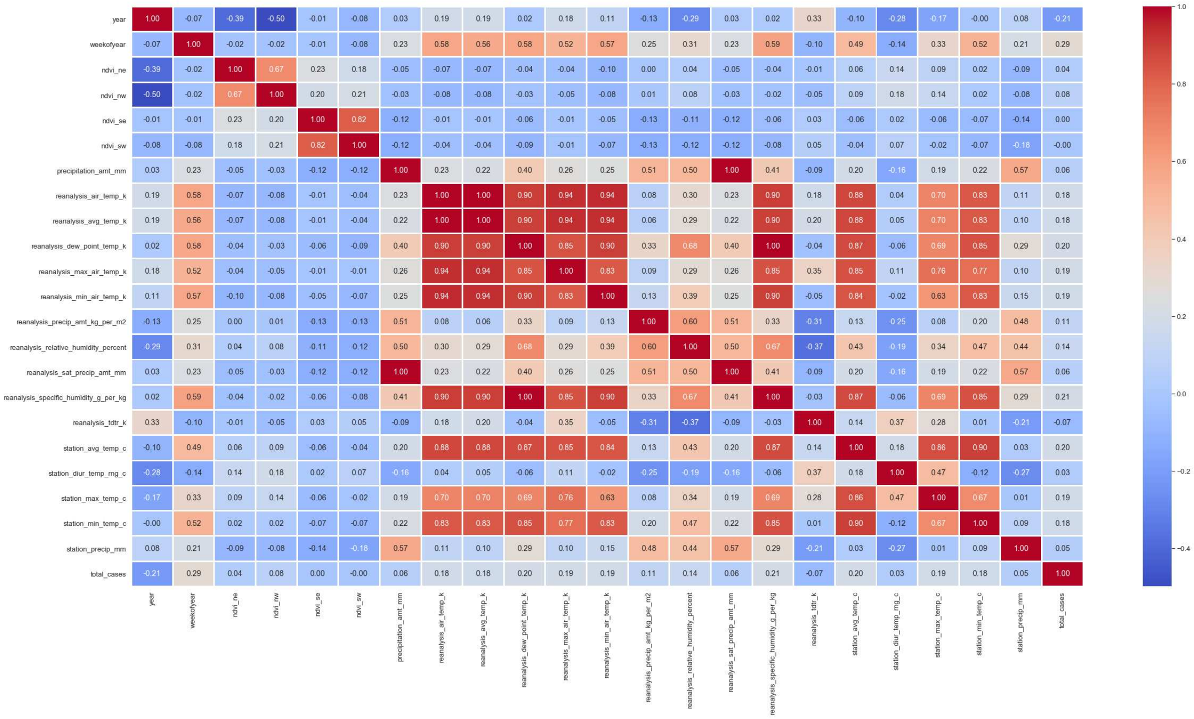
Now we check any correlation between features:

In [16]:
```
df.corr()
```

Out[16]:

	year	weekofyear	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precip
year	1.000000	-0.073143	-0.392312	-0.498367	-0.014863	-0.077928	
weekofyear	-0.073143	1.000000	-0.020271	-0.023549	-0.009380	-0.075804	
ndvi_ne	-0.392312	-0.020271	1.000000	0.673037	0.234049	0.177792	
ndvi_nw	-0.498367	-0.023549	0.673037	1.000000	0.196343	0.214615	
ndvi_se	-0.014863	-0.009380	0.234049	0.196343	1.000000	0.821354	
ndvi_sw	-0.077928	-0.075804	0.177792	0.214615	0.821354	1.000000	
precipitation_amt_mm	0.031612	0.231961	-0.048727	-0.032351	-0.119386	-0.118752	
reanalysis_air_temp_k	0.185813	0.575381	-0.073170	-0.077457	-0.014601	-0.043488	
reanalysis_avg_temp_k	0.189696	0.561001	-0.071176	-0.076375	-0.011905	-0.035999	
reanalysis_dew_point_temp_k	0.015428	0.578072	-0.040008	-0.026070	-0.062773	-0.087787	
reanalysis_max_air_temp_k	0.177772	0.519083	-0.044553	-0.046189	-0.007382	-0.014890	
reanalysis_min_air_temp_k	0.105355	0.574494	-0.096176	-0.075337	-0.045946	-0.072345	
reanalysis_precip_amt_kg_per_m2	-0.132494	0.253975	0.004448	0.009383	-0.130848	-0.126646	
reanalysis_relative_humidity_percent	-0.286206	0.306771	0.039138	0.077339	-0.114294	-0.118769	
reanalysis_sat_precip_amt_mm	0.031612	0.231961	-0.048727	-0.032351	-0.119386	-0.118752	
reanalysis_specific_humidity_g_per_kg	0.018129	0.585224	-0.035235	-0.020595	-0.058442	-0.080840	
reanalysis_tdtr_k	0.325336	-0.099084	-0.009248	-0.050657	0.029358	0.052465	
station_avg_temp_c	-0.097312	0.485038	0.064027	0.087298	-0.056545	-0.041495	
station_diur_temp_rng_c	-0.276963	-0.137093	0.142875	0.184124	0.018121	0.069843	
station_max_temp_c	-0.172907	0.325748	0.092365	0.136659	-0.063718	-0.017866	
station_min_temp_c	-0.002097	0.520129	0.018818	0.016428	-0.069275	-0.074045	
station_precip_mm	0.082920	0.213336	-0.085993	-0.076237	-0.140286	-0.175239	
total_cases	-0.212690	0.287134	0.037639	0.075307	0.001113	-0.000333	

```
In [17]: plt.figure(figsize=(40,20))
         sns.heatmap(df.corr(),cmap="coolwarm",annot=True,fmt='.2f',linewidths=2)
         plt.show()
```



Several features are highly correlated with each other:

- ndvi_se – Pixel southeast of city centroid
- ndvi_sw – Pixel southwest of city centroid
- ndvi_ne – Pixel northeast of city centroid
- ndvi_nw – Pixel northwest of city centroid
- reanalysis_air_temp_k – Mean air temperature
- reanalysis_avg_temp_k – Average air temperature
- reanalysis_dew_point_temp_k – Mean dew point temperature
- reanalysis_max_air_temp_k – Maximum air temperature
- reanalysis_min_air_temp_k – Minimum air temperature
- reanalysis_specific_humidity_g_per_kg – Mean specific humidity
- station_avg_temp_c – Average temperature
- station_max_temp_c – Maximum temperature
- station_min_temp_c – Minimum temperature

Data Preprocessing is next by checking missing values, preparing the data for modeling

Drop unwanted features

In [18]:
```
df.columns
```

Out[18]:
```
Index(['city', 'year', 'weekofyear', 'week_start_date', 'ndvi_ne', 'ndvi_nw',
       'ndvi_se', 'ndvi_sw', 'precipitation_amt_mm', 'reanalysis_air_temp_k',
       'reanalysis_avg_temp_k', 'reanalysis_dew_point_temp_k',
       'reanalysis_max_air_temp_k', 'reanalysis_min_air_temp_k',
       'reanalysis_precip_amt_kg_per_m2',
       'reanalysis_relative_humidity_percent', 'reanalysis_sat_precip_amt_mm',
       'reanalysis_specific_humidity_g_per_kg', 'reanalysis_tdtr_k',
       'station_avg_temp_c', 'station_diur_temp_rng_c', 'station_max_temp_c',
       'station_min_temp_c', 'station_precip_mm', 'total_cases'],
      dtype='object')
```

In [19]:
```
df.drop(['city', 'week_start_date', 'ndvi_ne', 'ndvi_nw', 'ndvi_se', 'ndvi_sw'],axi
s=1,inplace=True)
```

In [20]:
```
df
```

Out[20]:

	year	weekofyear	precipitation_amt_mm	reanalysis_air_temp_k	reanalysis_avg_temp_k	reanalysis_dew_p
0	1990	18	12.42	297.572857	297.742857	
1	1990	19	22.82	298.211429	298.442857	
2	1990	20	34.54	298.781429	298.878571	
3	1990	21	15.36	298.987143	299.228571	
4	1990	22	7.52	299.518571	299.664286	
...	...	...	...	...	...	
931	2008	13	27.19	296.958571	296.957143	
932	2008	14	3.82	298.081429	298.228571	
933	2008	15	16.96	297.460000	297.564286	
934	2008	16	0.00	297.630000	297.778571	
935	2008	17	0.00	298.672857	298.692857	

936 rows × 19 columns

Treat Missing Values

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
year                                     0
weekofyear                               0
precipitation_amt_mm                     9
reanalysis_air_temp_k                    6
reanalysis_avg_temp_k                    6
reanalysis_dew_point_temp_k              6
reanalysis_max_air_temp_k                6
reanalysis_min_air_temp_k                6
reanalysis_precip_amt_kg_per_m2          6
reanalysis_relative_humidity_percent     6
reanalysis_sat_precip_amt_mm             9
reanalysis_specific_humidity_g_per_kg    6
reanalysis_tdtr_k                        6
station_avg_temp_c                       6
station_diur_temp_rng_c                  6
station_max_temp_c                       6
station_min_temp_c                       6
station_precip_mm                        6
total_cases                              0
dtype: int64
```

In [22]:

```
df.dropna(inplace=True)
```

In [23]:

```
df.isnull().sum()
```

Out[23]:

```
year                                     0
weekofyear                               0
precipitation_amt_mm                     0
reanalysis_air_temp_k                    0
reanalysis_avg_temp_k                    0
reanalysis_dew_point_temp_k              0
reanalysis_max_air_temp_k                0
reanalysis_min_air_temp_k                0
reanalysis_precip_amt_kg_per_m2          0
reanalysis_relative_humidity_percent     0
reanalysis_sat_precip_amt_mm             0
reanalysis_specific_humidity_g_per_kg    0
reanalysis_tdtr_k                        0
station_avg_temp_c                       0
station_diur_temp_rng_c                  0
station_max_temp_c                       0
station_min_temp_c                       0
station_precip_mm                        0
total_cases                              0
dtype: int64
```

In [24]:

```
df
```

Out[24]:

	year	weekofyear	precipitation_amt_mm	reanalysis_air_temp_k	reanalysis_avg_temp_k	reanalysis_dew_p
0	1990	18	12.42	297.572857	297.742857	
1	1990	19	22.82	298.211429	298.442857	
2	1990	20	34.54	298.781429	298.878571	
3	1990	21	15.36	298.987143	299.228571	
4	1990	22	7.52	299.518571	299.664286	
...	...	...	...	...	...	
931	2008	13	27.19	296.958571	296.957143	
932	2008	14	3.82	298.081429	298.228571	
933	2008	15	16.96	297.460000	297.564286	
934	2008	16	0.00	297.630000	297.778571	
935	2008	17	0.00	298.672857	298.692857	

927 rows × 19 columns

Treat Duplicate Values

In [25]:

```
df.duplicated(keep='first').sum()
```

Out[25]:

```
0
```

Create and save processed dataset

In [26]:

```
df.to_csv("train.csv",index=False)
```

In [27]:

```
df = pd.read_csv("train.csv")
```

```
In [28]: df
```

Out[28]:

	year	weekofyear	precipitation_amt_mm	reanalysis_air_temp_k	reanalysis_avg_temp_k	reanalysis_dew_p
0	1990	18	12.42	297.572857	297.742857	
1	1990	19	22.82	298.211429	298.442857	
2	1990	20	34.54	298.781429	298.878571	
3	1990	21	15.36	298.987143	299.228571	
4	1990	22	7.52	299.518571	299.664286	
...	...	...	...	...	...	
922	2008	13	27.19	296.958571	296.957143	
923	2008	14	3.82	298.081429	298.228571	
924	2008	15	16.96	297.460000	297.564286	
925	2008	16	0.00	297.630000	297.778571	
926	2008	17	0.00	298.672857	298.692857	

927 rows × 19 columns

```
In [29]: df.shape

Out[29]: (927, 19)
```

Summary of training at least three linear regression models which should be variations that cover using a simple linear regression as a baseline, adding polynomial effects, and using a regularization regression. Preferably, all use the same training and test splits, or the same cross-validation method.

```
In [30]: X = df.iloc[:,0:18]
         y = df.iloc[:,18]
```

```
In [31]: X.values

Out[31]: array([[1.990e+03, 1.800e+01, 1.242e+01, ..., 2.940e+01, 2.000e+01,
                 1.600e+01],
                [1.990e+03, 1.900e+01, 2.282e+01, ..., 3.170e+01, 2.220e+01,
                 8.600e+00],
                [1.990e+03, 2.000e+01, 3.454e+01, ..., 3.220e+01, 2.280e+01,
                 4.140e+01],
                ...,
                [2.008e+03, 1.500e+01, 1.696e+01, ..., 2.940e+01, 2.170e+01,
                 3.070e+01],
                [2.008e+03, 1.600e+01, 0.000e+00, ..., 2.940e+01, 2.170e+01,
                 1.120e+01],
                [2.008e+03, 1.700e+01, 0.000e+00, ..., 3.170e+01, 2.330e+01,
                 3.000e-01]])
```

In [32]:

```
y.values
```

```
Out[32]: array([  4,   5,   4,   3,   6,   2,   4,   5,  10,   6,   8,   2,   6,
                 17,  23,  13,  21,  28,  24,  20,  40,  27,  42,  33,  43,  37,
                 57,  71,  44,  56,  53,  52,  47,  26,  27,  21,  21,  26,  34,
                 37,  17,  19,  25,  18,  21,  17,  17,  16,  16,  15,  23,  16,
                 17,  12,  17,  10,  15,  19,  21,  14,  18,  13,  14,  18,  23,
                 25,  62,  60,  76,  66,  64,  68,  89,  92, 140, 116, 142, 129,
                140, 140, 127, 129, 169, 141, 108,  78,  70, 104,  90,  85,  55,
                 53,  59,  40,  37,  29,  30,  30,  28,  23,  24,  29,  26,  23,
                 20,  19,  20,  26,  29,  31,  28,  26,  32,  35,  33,  30,  52,
                 59,  67,  65,  74,  70,  61,  53,  76,  61,  57,  44,  34,  47,
                 60,  60,  53,  36,  31,  32,  28,  33,  33,  35,  22,  13,  13,
                 21,  17,  11,   8,   8,   6,   6,   7,  12,  17,  10,  10,  18,
                 19,  12,  22,  12,  21,  18,  16,  16,  22,  17,  25,  23,  12,
                 25,  28,  27,  18,  23,  23,  29,  38,  36,  43,  46,  31,  25,
                 40,  31,  38,  30,  22,  31,  26,  35,  36,  39,  25,  31,  37,
                 33,  25,  24,  18,  23,  13,  18,  14,  17,  22,  13,  24,  31,
                 34,  31,  31,  38,  49,  42,  49,  55,  80,  84,  72,  89, 115,
                179, 202, 272, 302, 395, 426, 461, 381, 333, 353, 410, 364, 359,
                288, 221, 149, 112, 154,  91,  72,  56,  46,  37,  26,  17,  17,
                 20,  11,   7,  16,  14,  16,   5,   2,   6,   5,   4,   3,   4,
                 16,   8,   7,  10,  14,   7,   9,  11,  23,  17,  19,  24,  17,
                 28,  40,  33,  31,  33,  29,  30,  36,  48,  40,  28,  36,  19,
                 34,  23,  17,  17,  23,  14,  20,  13,  23,  20,  16,  16,  23,
                 14,  15,   4,   5,   5,  11,  11,   7,   4,   6,   5,   2,   4,
                  2,   4,   6,   6,   4,   6,  11,  16,   9,  12,  13,  27,  21,
                 19,  17,  24,  27,  30,  29,  25,  35,  33,  30,  29,  31,  29,
                 22,  27,  24,  26,  29,  22,  33,  24,  30,  20,  17,  24,  28,
                 18,  13,   9,  14,  11,  11,  19,  10,   8,   8,   9,   3,   7,
                 14,   4,   9,  14,   7,   9,   3,   3,  14,  12,  10,  21,  26,
                 47,  42,  31,  34,  33,  52,  56,  70, 112,  70,  47,  48,  49,
                 66,  56,  61,  67,  68,  49,  50,  56,  75,  63,  62,  41,  50,
                 34,  31,  38,  30,  32,  26,  30,  36,  35,  46,  48,  44,  51,
                 59,  71, 102, 128, 127, 150, 191, 256, 329, 263, 220, 204, 181,
                 99,  54,  80, 102, 127,  73,  68,  64,  55,  67,  84,  85,  67,
                 73,  89,  68,  56,  77,  75,  47,  50,  42,  28,  37,  37,  27,
                 12,  15,  22,   8,  15,  17,  10,   9,  11,  20,  13,  11,  16,
                 11,   7,  17,  14,  13,  15,  30,  25,  40,  44,  25,  21,  48,
                 56,  60,  45,  55,  32,  46,  61,  42,  37,  43,  34,  40,  25,
                 16,  17,  17,  16,  23,  18,  18,   9,   7,   7,   4,   3,   2,
                  8,   3,   1,   1,   2,   3,   3,   2,   0,   0,   2,   2,   0,
                  6,   3,   6,   2,   3,   2,   4,   5,   2,   9,   2,   4,   8,
                  6,   3,  11,  14,  15,  20,   9,  20,  28,  38,  30,  30,  23,
                 16,  22,  28,  14,  17,  20,  17,  10,  13,  20,   9,  18,   9,
                  8,  19,  11,   4,   6,   6,   8,  13,   8,   8,   5,  16,  12,
                 11,  18,  10,  22,  14,  16,  18,  27,  38,  35,  41,  51,  65,
                 55,  54,  62,  64,  56,  65,  71,  75,  71,  72,  47,  27,  35,
                 25,  19,  37,  38,  34,  26,  19,  18,  22,  16,  18,   6,  12,
                  6,   6,   3,   7,   6,   1,   3,   2,   2,   1,  10,   3,   3,
                  1,   1,   2,   6,   3,   3,   5,   4,   7,   6,   5,   7,   6,
                  4,   4,   7,   9,   5,   5,  10,   6,  13,   6,   5,   5,   9,
                  3,   6,  11,   7,   7,  15,   9,   6,   6,   6,   7,  10,   8,
                  7,  12,   3,   2,   7,   5,   5,   7,   7,   7,   7,  10,  13,
                 10,  14,  11,  20,  25,  17,  18,  25,  21,  31,  32,  26,  35,
                 28,  37,  41,  34,  30,  39,  39,  39,  34,  30,  37,  29,  26,
                 15,  22,  20,  14,  10,  21,  14,  14,   9,  11,   5,   6,   7,
                 11,   4,   3,   2,   6,  10,   7,   5,   3,  12,  13,  10,  13,
                 13,   8,  21,  18,   8,   7,  20,  14,  14,   7,  14,  10,  13,
                 27,  13,  18,  16,  16,  20,  17,   4,  15,   8,   6,  12,  15,
                 11,  15,  17,   7,   7,   8,   9,  12,  12,   5,   4,  11,   4,
                  5,   7,   1,   1,   4,   2,   6,   3,   4,  10,  12,  21,  26,
                 21,  30,  45,  56,  75,  83,  82, 126, 119, 137, 131, 112,  82,
                 73,  43,  55,  55,  53,  46,  43,  29,  22,  26,  13,  17,   8,
                 13,  10,  17,  19,   9,   9,   9,   3,   7,   7,   0,   2,   3,
                  3,   1,   3,   3,   3,   7,   3,   5,  11,   5,   5,   6,   6,
                  4,   4,   8,  14,  12,  16,  10,  16,  18,  15,  23,  17,  33,
                 15,  13,  11,  14,  17,  19,  20,  12,  21,   7,  19,  10,  13,
                 10,   8,  21,  11,   9,  14,  14,  15,  18,  16,  12,  20,   8,
                  3,  13,   4,   1,  10,   8,  13,  10,  21,  18,  21,  34,  25,
                 34,  33,  40,  42,  36,  72,  75,  76,  92,  71, 112, 106, 101,
                170, 135, 106,  68,  48,  48,  26,  33,  29,  17,  12,  13,  17,
                 15,  14,  15,  10,   9,   2,   6,   8,   5,   1,   2,   3,   4,
                  3,   1,   3,  51,  dtype=int64)
```

In [33]:
```
X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_size=
0.2, random_state=123)
```

In [34]:
```
X_train
```

Out[34]:
```
array([[2.003e+03, 1.600e+01, 5.960e+00, ..., 3.170e+01, 2.280e+01,
        4.650e+01],
       [2.004e+03, 4.000e+00, 0.000e+00, ..., 2.780e+01, 2.060e+01,
        1.020e+01],
       [2.006e+03, 2.200e+01, 6.492e+01, ..., 3.390e+01, 2.440e+01,
        2.210e+01],
       ...,
       [1.997e+03, 4.100e+01, 7.029e+01, ..., 3.330e+01, 2.330e+01,
        4.440e+01],
       [1.997e+03, 2.400e+01, 0.000e+00, ..., 3.280e+01, 2.390e+01,
        3.000e-01],
       [2.000e+03, 1.400e+01, 7.230e+00, ..., 3.060e+01, 2.170e+01,
        1.330e+01]])
```

In [35]:
```
X_test
```

Out[35]:
```
array([[1994.  ,   24.  ,   13.82, ...,   32.8 ,   24.4 ,    5.4 ],
       [1994.  ,    2.  ,    0.  , ...,   29.4 ,   21.1 ,   13.4 ],
       [1995.  ,   11.  ,   19.43, ...,   31.7 ,   21.1 ,   16.5 ],
       ...,
       [1994.  ,   43.  ,  111.52, ...,   32.8 ,   23.3 ,   39.2 ],
       [1998.  ,   45.  ,  121.65, ...,   31.1 ,   23.3 ,   81.5 ],
       [1991.  ,   25.  ,   44.25, ...,   33.9 ,   25.  ,    3.3 ]])
```

In [36]:
```
scaler =  StandardScaler()
```

In [37]:
```
X_train_scaled = scaler.fit_transform(X_train)
```

In [38]:
```
X_train_scaled
```

Out[38]:
```
array([[ 0.80231276, -0.72416376, -0.65577875, ...,  0.02951017,
         0.13318496,  0.64414028],
       [ 0.99615487, -1.53159499, -0.78691389, ..., -2.30058362,
        -1.33382934, -0.56126667],
       [ 1.38383909, -0.32044814,  0.64149106, ...,  1.34392205,
         1.20010444, -0.16610572],
       ...,
       [-0.36073991,  0.95798466,  0.7596447 , ...,  0.98544608,
         0.4665973 ,  0.57440599],
       [-0.36073991, -0.18587626, -0.78691389, ...,  0.68671611,
         0.8666921 , -0.89001402],
       [ 0.22078643, -0.85873563, -0.62783552, ..., -0.62769577,
        -0.60032219, -0.45832558]])
```

In [39]:
```
X_test_scaled = scaler.transform(X_test)
```

In [40]:

```
X_test_scaled
```

Out[40]:

```
array([[-0.94226624, -0.18587626, -0.48283878, ...,  0.68671611,
         1.20010444, -0.72065933],
       [-0.94226624, -1.66616687, -0.78691389, ..., -1.34464771,
        -1.000417  , -0.4550049 ],
       [-0.74842413, -1.06059344, -0.35940453, ...,  0.02951017,
        -1.000417  , -0.35206381],
       ...,
       [-0.94226624,  1.09255653,  1.66680946, ...,  0.68671611,
         0.4665973 ,  0.40173062],
       [-0.1668978 ,  1.2271284 ,  1.8896952 , ..., -0.3289658 ,
         0.4665973 ,  1.80637839],
       [-1.52379257, -0.11859033,  0.18669854, ...,  1.34392205,
         1.60019925, -0.79039361]])
```

In [41]:

```
y_train
```

Out[41]:

```
array([  7,  10,   5,  12,  22,   2,  33,  15,  18,  48,   5,  31,  29,
        12,  12,  13,  33,  14,   6,   4,  17,   9,   7,  16,  63,  14,
         8, 149,  21,  40,  14,  11,  91,  22,  30,  13,  18,  16,  17,
        20, 426,  19,   4,  15,  29,  49,  55,  25,  26,  39,   6,  25,
        41,  26,   3,  56,   2,  37,  42,  17,  10,  19,  62, 112,   7,
         7,  45,  10,  44,  26,  11,  15,   3,  53,   7,  72,  19,   7,
        28,  43,  80,  18,   2,  14, 353,  25,   7,   2,  10,  22, 364,
         7,  11,  53,   6,   2,  55, 169,  23,   8,  30,  11,  16,  65,
        20,  47,  14,  37,  28,  99,  34,  33,  15,  70,  21,  12,  59,
         3, 220,   1,  17,  84,   4,  20,   7,  12,  16,  76,  68,   6,
        25,  24,  24, 170,  27,  26,  50,  16,  12,  23,  31,   2,   2,
        27,  17,  71,  28,   7,   5,   6,  21,  21,  74,  14,  44, 263,
        39, 127,   4, 106,  70,  60,  25,  42,  25,  21,  29,  20,  61,
        82,  15,   5,  49,  62,  36,  36, 108,  46,  18,  25,   7,  11,
        21,   2,  29, 381,  54, 137,  18,   2,  13,  23,  37,   9,  13,
         5,  17,  35, 204,   6,  47,  10,  17,  13,  89,  10,   6,  20,
        17,   6,  51,  43,  30,   2,  57,   3,  23,  18,  35,   1,  48,
        26,  18,   3,   8,  67,  13,   2,  25,  26,   3,  23,  12,  34,
         4,  13,  40,  32,   9,  13, 119,  10,  52,   7,  26,   8,   3,
        75,  24,  33,  31,  53,   7,   4,  89,   9,  25,   3,  68,  11,
        71,   4,  30,   5,  77,  47,  12,   3,  20,  34,  43,   7,  17,
        23,  13,   9,   1,  36, 101,   3,  10,   4,   3, 106,  40,  14,
         8,  40,  30,   6,  20,  35,  31,   9,  40,  19,   6,  42,  10,
        61,  19,  68,  33, 288,  20,   2,   5,  15,  13,  50,   2,   3,
         6,  15,  37,  70,   5,  29,  46,  18,  12,   3,  21,  72, 410,
         9,  42,  16,   9,  27,  17,  13,  16,  38,  61,  11,  17,  40,
         5,  33,   8,   5,  75,  15,  38,  19,  85,  31,  39,  20,   3,
         8,  30,  29,  10,  71, 272,  80,  11,  35,  56,  64,   5,   6,
        17,   2,   7,  89, 359,  44,  17,   1,   5,   9,   2,  19,  29,
        10,   6,  33,  33,   6,  20,  34,   9,   6,  12,  56,   4,   6,
        34,   8,  14,  14,  12,  13,   4,  56,  14,  26,   4,  11,  17,
        34,   1,  17,   7,  62,   9,   6,  13,   6,  84,  34,  17,   8,
        71,  48,  14,   7,  44,  37,  56, 129,   9,  36,   5,  26,  20,
        56,  30,   9,  92,  51,  28, 141,  17,  29,  12,   5,  23,  64,
        42,  16, 126,  26,   3,  11,   3,  82, 256,  13,   6,   0,  72,
        75,  16,   7,  20,  21,  57,  17, 154,  75,   3,  11,   9,  20,
        18,  29,  56,  29,  33,  19,  21,  38,  38,  14,   2,  21,   3,
        12,  33, 191, 115, 395,  27,   0,  56,   6,  27,  11,   7,  28,
        28,  14,  30,  12,   8,  11,   8,  13,   1,   8,  76,  20,  14,
        31,  67,   6,  66,   7,  37,   9,  11,  14,  32,  19,   6,   2,
       102,   7,  26,   6,   3,   8,   3,  67,  48,  32,  31,  73,   5,
        18,  15,  59,  36,   6,  21,  10,  17,   7,  32,  25,  14,  18,
         6,  60,  23,   4,   6,  24,  55,  26,   2,  11,  56,  22,  14,
        85,   5,  17,   4, 181,  40,  16,   7,  43,   3,  17, 142,  17,
         9,   0,  25,  19,  24,  10, 129,  22,  21,  12,  13, 329,  47,
        48,  17,   4,  71,  16,  14,  46,  13,  90,   5,  27,   4,  12,
        18,  46, 112, 128,   7, 127,   4,  29,  16,   6,  34,  17,  70,
        31,   8,  13,  21,  25,  31,  34,   3,  41,  13,   9,  67,  22,
        18,  16,  23, 104,   7,   7, 202,  15,   4,   3,  14,  37,   6,
        15,  10,  35,  60,   0,  29,  15,  11,  16,  16,   3,  13,   2,
        21,  11,   7,  73,  15,  39,  19,  22,  20,  47,   8,  26,   6,
        45,  66,  10,  10,   3,   3,  41,  21,  22,  11,  10,  24,  55,
         8,   3, 131,  68,   5, 102,  17,  14,  16,   5,  25,  64,  23,
        35,   9,   5,  42,  15,  10,  31,  21,  34,  23,   4,  32,  38,
         4,   8,  27,   3,  37,  24,   4,  28,   3,   2,  76,  34,  18,
       127,   5,  13,  27,  30,  26, 302,  30,  92,  16,   4,  35,  30,
         7,  49,   5,  53,  20,  47,  28,  13,  28,  13,  52,   4,   1],
      dtype=int64)
```

Simple Linear Regression

In [42]:

```
lr = LinearRegression()
```

```
In [43]: lr.fit(X_train_scaled, y_train)

Out[43]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
In [44]: lr.coef_

Out[44]: array([ -13.80623407,    8.99915311,   -1.36325657, -175.84478333,
                 -32.74754264,  209.53405334,   20.4104613 ,    4.2651429 ,
                   3.28183336, -123.699823  ,   -1.36325657,   54.25518557,
                  -2.69328076,   -9.40856536,   -2.59603833,    4.96282398,
                  -1.77749875,    0.39295523])
```

```
In [45]: lr.intercept_

Out[45]: 34.546558704402564
```

```
In [46]: ypred_lr = lr.predict(X_test_scaled)
```

```
In [47]: y_test[:10]

Out[47]: array([31, 26,  7,  5, 65,  3, 23,  1, 78,  4], dtype=int64)
```

```
In [48]: ypred_lr[:10]

Out[48]: array([ 36.55828463,  17.31594432,  39.84648332,  36.74135096,
                 42.85447406, -10.5994066 ,  10.5371052 ,  -0.2688629 ,
                 66.21352125,  37.09086605])
```

Linear Regression Model Evaluation

```
In [49]: mse = mean_squared_error(y_test,ypred_lr)
         mse

Out[49]: 2296.1327558860244
```

```
In [50]: rmse = np.sqrt(mse)
         rmse

Out[50]: 47.91797946372556
```

```
In [51]: r2_score(y_test,ypred_lr)

Out[51]: 0.11917629922609252
```

Create Polynomial Features and Train Model

```
In [52]: pf = PolynomialFeatures(degree=3, include_bias=False)
```

```
In [53]: X_pf = pf.fit_transform(X)
In [54]: X_pf
```

```
Out[54]: array([[1.9900000e+03, 1.8000000e+01, 1.2420000e+01, ..., 6.4000000e+03,
                 5.1200000e+03, 4.0960000e+03],
                [1.9900000e+03, 1.9000000e+01, 2.2820000e+01, ..., 4.2384240e+03,
                 1.6419120e+03, 6.3605600e+02],
                [1.9900000e+03, 2.0000000e+01, 3.4540000e+01, ..., 2.1521376e+04,
                 3.9078288e+04, 7.0957944e+04],
                ...,
                [2.0080000e+03, 1.5000000e+01, 1.6960000e+01, ..., 1.4456323e+04,
                 2.0452033e+04, 2.8934443e+04],
                [2.0080000e+03, 1.6000000e+01, 0.0000000e+00, ..., 5.2739680e+03,
                 2.7220480e+03, 1.4049280e+03],
                [2.0080000e+03, 1.7000000e+01, 0.0000000e+00, ..., 1.6286700e+02,
                 2.0970000e+00, 2.7000000e-02]])
```

```
In [55]: X_train, X_test, y_train, y_test = train_test_split(X_pf, y.values, test_size=0.2,
         random_state=123)
```

```
In [56]: X_train
```

```
Out[56]: array([[2.00300000e+03, 1.60000000e+01, 5.96000000e+00, ...,
                 2.41725600e+04, 4.92993000e+04, 1.00544625e+05],
                [2.00400000e+03, 4.00000000e+00, 0.00000000e+00, ...,
                 4.32847200e+03, 2.14322400e+03, 1.06120800e+03],
                [2.00600000e+03, 2.20000000e+01, 6.49200000e+01, ...,
                 1.31574560e+04, 1.19172040e+04, 1.07938610e+04],
                ...,
                [1.99700000e+03, 4.10000000e+01, 7.02900000e+01, ...,
                 2.41043160e+04, 4.59326880e+04, 8.75283840e+04],
                [1.99700000e+03, 2.40000000e+01, 0.00000000e+00, ...,
                 1.71363000e+02, 2.15100000e+00, 2.70000000e-02],
                [2.00000000e+03, 1.40000000e+01, 7.23000000e+00, ...,
                 6.26283700e+03, 3.83851300e+03, 2.35263700e+03]])
```

```
In [57]: X_test
```

```
Out[57]: array([[1.99400000e+03, 2.40000000e+01, 1.38200000e+01, ...,
                 3.21494400e+03, 7.11504000e+02, 1.57464000e+02],
                [1.99400000e+03, 2.00000000e+00, 0.00000000e+00, ...,
                 5.96581400e+03, 3.78871600e+03, 2.40610400e+03],
                [1.99500000e+03, 1.10000000e+01, 1.94300000e+01, ...,
                 7.34596500e+03, 5.74447500e+03, 4.49212500e+03],
                ...,
                [1.99400000e+03, 4.30000000e+01, 1.11520000e+02, ...,
                 2.12812880e+04, 3.58037120e+04, 6.02362880e+04],
                [1.99800000e+03, 4.50000000e+01, 1.21650000e+02, ...,
                 4.42455350e+04, 1.54764425e+05, 5.41343375e+05],
                [1.99100000e+03, 2.50000000e+01, 4.42500000e+01, ...,
                 2.06250000e+03, 2.72250000e+02, 3.59370000e+01]])
```

```
In [58]: scaler =  StandardScaler()
```

```
In [59]: X_train_scaled = scaler.fit_transform(X_train)
```

In [60]:
```
X_train_scaled
```

Out[60]:
```
array([[ 0.80231276, -0.72416376, -0.65577875, ...,  0.645733  ,
         0.10888467, -0.05782687],
       [ 0.99615487, -1.53159499, -0.78691389, ..., -0.62078406,
        -0.31891229, -0.14300692],
       [ 1.38383909, -0.32044814,  0.64149106, ..., -0.05728833,
        -0.23024335, -0.1346736 ],
       ...,
       [-0.36073991,  0.95798466,  0.7596447 , ...,  0.64137743,
         0.07834298, -0.06897168],
       [-0.36073991, -0.18587626, -0.78691389, ..., -0.88610487,
        -0.33833597, -0.14391553],
       [ 0.22078643, -0.85873563, -0.62783552, ..., -0.49732632,
        -0.30353273, -0.14190117]])
```

In [61]:
```
X_test_scaled = scaler.transform(X_test)
```

In [62]:
```
X_test_scaled
```

Out[62]:
```
array([[-0.94226624, -0.18587626, -0.48283878, ..., -0.6918532 ,
        -0.33190076, -0.14378073],
       [-0.94226624, -1.66616687, -0.78691389, ..., -0.51628334,
        -0.30398449, -0.14185539],
       [-0.74842413, -1.06059344, -0.35940453, ..., -0.42819742,
        -0.28624196, -0.14006929],
       ...,
       [-0.94226624,  1.09255653,  1.66680946, ...,  0.4612022 ,
        -0.01354646, -0.09233982],
       [-0.1668978 ,  1.2271284 ,  1.8896952 , ...,  1.9268584 ,
         1.06565765,  0.31959546],
       [-1.52379257, -0.11859033,  0.18669854, ..., -0.76540609,
        -0.33588564, -0.14388478]])
```

In [63]:
```
y_train
```

Out[63]:
```
array([  7,  10,   5,  12,  22,   2,  33,  15,  18,  48,   5,  31,  29,
        12,  12,  13,  33,  14,   6,   4,  17,   9,   7,  16,  63,  14,
         8, 149,  21,  40,  14,  11,  91,  22,  30,  13,  18,  16,  17,
        20, 426,  19,   4,  15,  29,  49,  55,  25,  26,  39,   6,  25,
        41,  26,   3,  56,   2,  37,  42,  17,  10,  19,  62, 112,   7,
         7,  45,  10,  44,  26,  11,  15,   3,  53,   7,  72,  19,   7,
        28,  43,  80,  18,   2,  14, 353,  25,   7,   2,  10,  22, 364,
         7,  11,  53,   6,   2,  55, 169,  23,   8,  30,  11,  16,  65,
        20,  47,  14,  37,  28,  99,  34,  33,  15,  70,  21,  12,  59,
         3, 220,   1,  17,  84,   4,  20,   7,  12,  16,  76,  68,   6,
        25,  24,  24, 170,  27,  26,  50,  16,  12,  23,  31,   2,   2,
        27,  17,  71,  28,   7,   5,   6,  21,  21,  74,  14,  44, 263,
        39, 127,   4, 106,  70,  60,  25,  42,  25,  21,  29,  20,  61,
        82,  15,   5,  49,  62,  36,  36, 108,  46,  18,  25,   7,  11,
        21,   2,  29, 381,  54, 137,  18,   2,  13,  23,  37,   9,  13,
         5,  17,  35, 204,   6,  47,  10,  17,  13,  89,  10,   6,  20,
        17,   6,  51,  43,  30,   2,  57,   3,  23,  18,  35,   1,  48,
        26,  18,   3,   8,  67,  13,   2,  25,  26,   3,  23,  12,  34,
         4,  13,  40,  32,   9,  13, 119,  10,  52,   7,  26,   8,   3,
        75,  24,  33,  31,  53,   7,   4,  89,   9,  25,   3,  68,  11,
        71,   4,  30,   5,  77,  47,  12,   3,  20,  34,  43,   7,  17,
        23,  13,   9,   1,  36, 101,   3,  10,   4,   3, 106,  40,  14,
         8,  40,  30,   6,  20,  35,  31,   9,  40,  19,   6,  42,  10,
        61,  19,  68,  33, 288,  20,   2,   5,  15,  13,  50,   2,   3,
         6,  15,  37,  70,   5,  29,  46,  18,  12,   3,  21,  72, 410,
         9,  42,  16,   9,  27,  17,  13,  16,  38,  61,  11,  17,  40,
         5,  33,   8,   5,  75,  15,  38,  19,  85,  31,  39,  20,   3,
         8,  30,  29,  10,  71, 272,  80,  11,  35,  56,  64,   5,   6,
        17,   2,   7,  89, 359,  44,  17,   1,   5,   9,   2,  19,  29,
        10,   6,  33,  33,   6,  20,  34,   9,   6,  12,  56,   4,   6,
        34,   8,  14,  14,  12,  13,   4,  56,  14,  26,   4,  11,  17,
        34,   1,  17,   7,  62,   9,   6,  13,   6,  84,  34,  17,   8,
        71,  48,  14,   7,  44,  37,  56, 129,   9,  36,   5,  26,  20,
        56,  30,   9,  92,  51,  28, 141,  17,  29,  12,   5,  23,  64,
        42,  16, 126,  26,   3,  11,   3,  82, 256,  13,   6,   0,  72,
        75,  16,   7,  20,  21,  57,  17, 154,  75,   3,  11,   9,  20,
        18,  29,  56,  29,  33,  19,  21,  38,  38,  14,   2,  21,   3,
        12,  33, 191, 115, 395,  27,   0,  56,   6,  27,  11,   7,  28,
        28,  14,  30,  12,   8,  11,   8,  13,   1,   8,  76,  20,  14,
        31,  67,   6,  66,   7,  37,   9,  11,  14,  32,  19,   6,   2,
       102,   7,  26,   6,   3,   8,   3,  67,  48,  32,  31,  73,   5,
        18,  15,  59,  36,   6,  21,  10,  17,   7,  32,  25,  14,  18,
         6,  60,  23,   4,   6,  24,  55,  26,   2,  11,  56,  22,  14,
        85,   5,  17,   4, 181,  40,  16,   7,  43,   3,  17, 142,  17,
         9,   0,  25,  19,  24,  10, 129,  22,  21,  12,  13, 329,  47,
        48,  17,   4,  71,  16,  14,  46,  13,  90,   5,  27,   4,  12,
        18,  46, 112, 128,   7, 127,   4,  29,  16,   6,  34,  17,  70,
        31,   8,  13,  21,  25,  31,  34,   3,  41,  13,   9,  67,  22,
        18,  16,  23, 104,   7,   7, 202,  15,   4,   3,  14,  37,   6,
        15,  10,  35,  60,   0,  29,  15,  11,  16,  16,   3,  13,   2,
        21,  11,   7,  73,  15,  39,  19,  22,  20,  47,   8,  26,   6,
        45,  66,  10,  10,   3,   3,  41,  21,  22,  11,  10,  24,  55,
         8,   3, 131,  68,   5, 102,  17,  14,  16,   5,  25,  64,  23,
        35,   9,   5,  42,  15,  10,  31,  21,  34,  23,   4,  32,  38,
         4,   8,  27,   3,  37,  24,   4,  28,   3,   2,  76,  34,  18,
       127,   5,  13,  27,  30,  26, 302,  30,  92,  16,   4,  35,  30,
         7,  49,   5,  53,  20,  47,  28,  13,  28,  13,  52,   4,   1],
      dtype=int64)
```

Simple Linear Regression (Polynomial)

In [64]:
```
lrpf = LinearRegression()
```

```
In [65]: lrpf.fit(X_train_scaled, y_train)

Out[65]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
In [66]: lrpf.coef_

Out[66]: array([ 4.04497778e+05, -2.55668313e+05,  6.50974813e+04, ...,
                -4.21015756e+03,  5.73243989e+03, -5.20813936e+00])
```

```
In [67]: lrpf.intercept_

Out[67]: 34.54655886532953
```

```
In [68]: ypred_lrpf = lrpf.predict(X_test_scaled)
```

```
In [69]: y_test[:10]

Out[69]: array([31, 26,  7,  5, 65,  3, 23,  1, 78,  4], dtype=int64)
```

```
In [70]: ypred_lrpf[:10]

Out[70]: array([  27.22557408, -221.18226368, -149.19290461, 1216.07131533,
                  14.47456172,   58.55744448,  -39.79681531, -110.82577654,
                 111.5734072 ,  -18.22927902])
```

Linear Regression Model Evaluation for Polynomial

```
In [71]: mse = mean_squared_error(y_test,ypred_lrpf)
         mse

Out[71]: 592089.1907837309
```

```
In [72]: rmse = np.sqrt(mse)
         rmse

Out[72]: 769.4733203846192
```

```
In [73]: r2_score(y_test,ypred_lrpf)

Out[73]: -226.13242118838605
```

Ridge Regression

In [74]:

```
# Reload the original df again
df
```

Out[74]:

	year	weekofyear	precipitation_amt_mm	reanalysis_air_temp_k	reanalysis_avg_temp_k	reanalysis_dew_p
0	1990	18	12.42	297.572857	297.742857	
1	1990	19	22.82	298.211429	298.442857	
2	1990	20	34.54	298.781429	298.878571	
3	1990	21	15.36	298.987143	299.228571	
4	1990	22	7.52	299.518571	299.664286	
...	...	...	...	...	...	
922	2008	13	27.19	296.958571	296.957143	
923	2008	14	3.82	298.081429	298.228571	
924	2008	15	16.96	297.460000	297.564286	
925	2008	16	0.00	297.630000	297.778571	
926	2008	17	0.00	298.672857	298.692857	

927 rows × 19 columns

In [75]:

```
X = df.iloc[:,0:18]
y = df.iloc[:,18]
```

In [76]:

```
X.values
```

Out[76]:

```
array([[1.990e+03, 1.800e+01, 1.242e+01, ..., 2.940e+01, 2.000e+01,
        1.600e+01],
       [1.990e+03, 1.900e+01, 2.282e+01, ..., 3.170e+01, 2.220e+01,
        8.600e+00],
       [1.990e+03, 2.000e+01, 3.454e+01, ..., 3.220e+01, 2.280e+01,
        4.140e+01],
       ...,
       [2.008e+03, 1.500e+01, 1.696e+01, ..., 2.940e+01, 2.170e+01,
        3.070e+01],
       [2.008e+03, 1.600e+01, 0.000e+00, ..., 2.940e+01, 2.170e+01,
        1.120e+01],
       [2.008e+03, 1.700e+01, 0.000e+00, ..., 3.170e+01, 2.330e+01,
        3.000e-01]])
```

In [77]:

```
y.values
```

```
Out[77]: array([  4,   5,   4,   3,   6,   2,   4,   5,  10,   6,   8,   2,   6,
                 17,  23,  13,  21,  28,  24,  20,  40,  27,  42,  33,  43,  37,
                 57,  71,  44,  56,  53,  52,  47,  26,  27,  21,  21,  26,  34,
                 37,  17,  19,  25,  18,  21,  17,  17,  16,  16,  15,  23,  16,
                 17,  12,  17,  10,  15,  19,  21,  14,  18,  13,  14,  18,  23,
                 25,  62,  60,  76,  66,  64,  68,  89,  92, 140, 116, 142, 129,
                140, 140, 127, 129, 169, 141, 108,  78,  70, 104,  90,  85,  55,
                 53,  59,  40,  37,  29,  30,  30,  28,  23,  24,  29,  26,  23,
                 20,  19,  20,  26,  29,  31,  28,  26,  32,  35,  33,  30,  52,
                 59,  67,  65,  74,  70,  61,  53,  76,  61,  57,  44,  34,  47,
                 60,  60,  53,  36,  31,  32,  28,  33,  33,  35,  22,  13,  13,
                 21,  17,  11,   8,   8,   6,   6,   7,  12,  17,  10,  10,  18,
                 19,  12,  22,  12,  21,  18,  16,  16,  22,  17,  25,  23,  12,
                 25,  28,  27,  18,  23,  23,  29,  38,  36,  43,  46,  31,  25,
                 40,  31,  38,  30,  22,  31,  26,  35,  36,  39,  25,  31,  37,
                 33,  25,  24,  18,  23,  13,  18,  14,  17,  22,  13,  24,  31,
                 34,  31,  31,  38,  49,  42,  49,  55,  80,  84,  72,  89, 115,
                179, 202, 272, 302, 395, 426, 461, 381, 333, 353, 410, 364, 359,
                288, 221, 149, 112, 154,  91,  72,  56,  46,  37,  26,  17,  17,
                 20,  11,   7,  16,  14,  16,   5,   2,   6,   5,   4,   3,   4,
                 16,   8,   7,  10,  14,   7,   9,  11,  23,  17,  19,  24,  17,
                 28,  40,  33,  31,  33,  29,  30,  36,  48,  40,  28,  36,  19,
                 34,  23,  17,  17,  23,  14,  20,  13,  23,  20,  16,  16,  23,
                 14,  15,   4,   5,   5,  11,  11,   7,   4,   6,   5,   2,   4,
                  2,   4,   6,   6,   4,   6,  11,  16,   9,  12,  13,  27,  21,
                 19,  17,  24,  27,  30,  29,  25,  35,  33,  30,  29,  31,  29,
                 22,  27,  24,  26,  29,  22,  33,  24,  30,  20,  17,  24,  28,
                 18,  13,   9,  14,  11,  11,  19,  10,   8,   8,   9,   3,   7,
                 14,   4,   9,  14,   7,   9,   3,   3,  14,  12,  10,  21,  26,
                 47,  42,  31,  34,  33,  52,  56,  70, 112,  70,  47,  48,  49,
                 66,  56,  61,  67,  68,  49,  50,  56,  75,  63,  62,  41,  50,
                 34,  31,  38,  30,  32,  26,  30,  36,  35,  46,  48,  44,  51,
                 59,  71, 102, 128, 127, 150, 191, 256, 329, 263, 220, 204, 181,
                 99,  54,  80, 102, 127,  73,  68,  64,  55,  67,  84,  85,  67,
                 73,  89,  68,  56,  77,  75,  47,  50,  42,  28,  37,  37,  27,
                 12,  15,  22,   8,  15,  17,  10,   9,  11,  20,  13,  11,  16,
                 11,   7,  17,  14,  13,  15,  30,  25,  40,  44,  25,  21,  48,
                 56,  60,  45,  55,  32,  46,  61,  42,  37,  43,  34,  40,  25,
                 16,  17,  17,  16,  23,  18,  18,   9,   7,   7,   4,   3,   2,
                  8,   3,   1,   1,   2,   3,   3,   2,   0,   0,   2,   2,   0,
                  6,   3,   6,   2,   3,   2,   4,   5,   2,   9,   2,   4,   8,
                  6,   3,  11,  14,  15,  20,   9,  20,  28,  38,  30,  30,  23,
                 16,  22,  28,  14,  17,  20,  17,  10,  13,  20,   9,  18,   9,
                  8,  19,  11,   4,   6,   6,   8,  13,   8,   8,   5,  16,  12,
                 11,  18,  10,  22,  14,  16,  18,  27,  38,  35,  41,  51,  65,
                 55,  54,  62,  64,  56,  65,  71,  75,  71,  72,  47,  27,  35,
                 25,  19,  37,  38,  34,  26,  19,  18,  22,  16,  18,   6,  12,
                  6,   6,   3,   7,   6,   1,   3,   2,   2,   1,  10,   3,   3,
                  1,   1,   2,   6,   3,   3,   5,   4,   7,   6,   5,   7,   6,
                  4,   4,   7,   9,   5,   5,  10,   6,  13,   6,   5,   5,   9,
                  3,   6,  11,   7,   7,  15,   9,   6,   6,   6,   7,  10,   8,
                  7,  12,   3,   2,   7,   5,   5,   7,   7,   7,   7,  10,  13,
                 10,  14,  11,  20,  25,  17,  18,  25,  21,  31,  32,  26,  35,
                 28,  37,  41,  34,  30,  39,  39,  39,  34,  30,  37,  29,  26,
                 15,  22,  20,  14,  10,  21,  14,  14,   9,  11,   5,   6,   7,
                 11,   4,   3,   2,   6,  10,   7,   5,   3,  12,  13,  10,  13,
                 13,   8,  21,  18,   8,   7,  20,  14,  14,   7,  14,  10,  13,
                 27,  13,  18,  16,  16,  20,  17,   4,  15,   8,   6,  12,  15,
                 11,  15,  17,   7,   7,   8,   9,  12,  12,   5,   4,  11,   4,
                  5,   7,   1,   1,   4,   2,   6,   3,   4,  10,  12,  21,  26,
                 21,  30,  45,  56,  75,  83,  82, 126, 119, 137, 131, 112,  82,
                 73,  43,  55,  55,  53,  46,  43,  29,  22,  26,  13,  17,   8,
                 13,  10,  17,  19,   9,   9,   9,   3,   7,   7,   0,   2,   3,
                  3,   1,   3,   3,   3,   7,   3,   5,  11,   5,   5,   6,   6,
                  4,   4,   8,  14,  12,  16,  10,  16,  18,  15,  23,  17,  33,
                 15,  13,  11,  14,  17,  19,  20,  12,  21,   7,  19,  10,  13,
                 10,   8,  21,  11,   9,  14,  14,  15,  18,  16,  12,  20,   8,
                  3,   13,    4,    1,   10,    8,   13,   10,   21,   18,   21,   34,   25,
                 34,   33,   40,   42,   36,   72,   75,   76,   92,   71,  112,  106,  101,
                170,  135,  106,   68,   48,   48,   26,   33,   29,   17,   12,   13,   17,
                 15,   14,   15,   10,    9,    2,    6,    8,    5,    1,    2,    3,    4,
                  3,    1,    3,   51], dtype=int64)
```

```
In [78]: X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_size=
         0.2, random_state=123)
```

```
In [79]: X_train
```

```
Out[79]: array([[2.003e+03, 1.600e+01, 5.960e+00, ..., 3.170e+01, 2.280e+01,
                 4.650e+01],
                [2.004e+03, 4.000e+00, 0.000e+00, ..., 2.780e+01, 2.060e+01,
                 1.020e+01],
                [2.006e+03, 2.200e+01, 6.492e+01, ..., 3.390e+01, 2.440e+01,
                 2.210e+01],
                ...,
                [1.997e+03, 4.100e+01, 7.029e+01, ..., 3.330e+01, 2.330e+01,
                 4.440e+01],
                [1.997e+03, 2.400e+01, 0.000e+00, ..., 3.280e+01, 2.390e+01,
                 3.000e-01],
                [2.000e+03, 1.400e+01, 7.230e+00, ..., 3.060e+01, 2.170e+01,
                 1.330e+01]])
```

```
In [80]: X_test
```

```
Out[80]: array([[1994.  ,   24.  ,   13.82, ...,   32.8 ,   24.4 ,    5.4 ],
                [1994.  ,    2.  ,    0.  , ...,   29.4 ,   21.1 ,   13.4 ],
                [1995.  ,   11.  ,   19.43, ...,   31.7 ,   21.1 ,   16.5 ],
                ...,
                [1994.  ,   43.  ,  111.52, ...,   32.8 ,   23.3 ,   39.2 ],
                [1998.  ,   45.  ,  121.65, ...,   31.1 ,   23.3 ,   81.5 ],
                [1991.  ,   25.  ,   44.25, ...,   33.9 ,   25.  ,    3.3 ]])
```

```
In [81]: scaler =  StandardScaler()
```

```
In [82]: X_train_scaled = scaler.fit_transform(X_train)
```

```
In [83]: X_train_scaled
```

```
Out[83]: array([[ 0.80231276, -0.72416376, -0.65577875, ...,  0.02951017,
                  0.13318496,  0.64414028],
                [ 0.99615487, -1.53159499, -0.78691389, ..., -2.30058362,
                 -1.33382934, -0.56126667],
                [ 1.38383909, -0.32044814,  0.64149106, ...,  1.34392205,
                  1.20010444, -0.16610572],
                ...,
                [-0.36073991,  0.95798466,  0.7596447 , ...,  0.98544608,
                  0.4665973 ,  0.57440599],
                [-0.36073991, -0.18587626, -0.78691389, ...,  0.68671611,
                  0.8666921 , -0.89001402],
                [ 0.22078643, -0.85873563, -0.62783552, ..., -0.62769577,
                 -0.60032219, -0.45832558]])
```

```
In [84]: X_test_scaled = scaler.transform(X_test)
In [85]: X_test_scaled
```

```
Out[85]: array([[-0.94226624, -0.18587626, -0.48283878, ...,  0.68671611,
                  1.20010444, -0.72065933],
                [-0.94226624, -1.66616687, -0.78691389, ..., -1.34464771,
                 -1.000417  , -0.4550049 ],
                [-0.74842413, -1.06059344, -0.35940453, ...,  0.02951017,
                 -1.000417  , -0.35206381],
                ...,
                [-0.94226624,  1.09255653,  1.66680946, ...,  0.68671611,
                  0.4665973 ,  0.40173062],
                [-0.1668978 ,  1.2271284 ,  1.8896952 , ..., -0.3289658 ,
                  0.4665973 ,  1.80637839],
                [-1.52379257, -0.11859033,  0.18669854, ...,  1.34392205,
                  1.60019925, -0.79039361]])
```

```
In [86]: y_train
```

```
Out[86]: array([  7,  10,   5,  12,  22,   2,  33,  15,  18,  48,   5,  31,  29,
                12,  12,  13,  33,  14,   6,   4,  17,   9,   7,  16,  63,  14,
                 8, 149,  21,  40,  14,  11,  91,  22,  30,  13,  18,  16,  17,
                20, 426,  19,   4,  15,  29,  49,  55,  25,  26,  39,   6,  25,
                41,  26,   3,  56,   2,  37,  42,  17,  10,  19,  62, 112,   7,
                 7,  45,  10,  44,  26,  11,  15,   3,  53,   7,  72,  19,   7,
                28,  43,  80,  18,   2,  14, 353,  25,   7,   2,  10,  22, 364,
                 7,  11,  53,   6,   2,  55, 169,  23,   8,  30,  11,  16,  65,
                20,  47,  14,  37,  28,  99,  34,  33,  15,  70,  21,  12,  59,
                 3, 220,   1,  17,  84,   4,  20,   7,  12,  16,  76,  68,   6,
                25,  24,  24, 170,  27,  26,  50,  16,  12,  23,  31,   2,   2,
                27,  17,  71,  28,   7,   5,   6,  21,  21,  74,  14,  44, 263,
                39, 127,   4, 106,  70,  60,  25,  42,  25,  21,  29,  20,  61,
                82,  15,   5,  49,  62,  36,  36, 108,  46,  18,  25,   7,  11,
                21,   2,  29, 381,  54, 137,  18,   2,  13,  23,  37,   9,  13,
                 5,  17,  35, 204,   6,  47,  10,  17,  13,  89,  10,   6,  20,
                17,   6,  51,  43,  30,   2,  57,   3,  23,  18,  35,   1,  48,
                26,  18,   3,   8,  67,  13,   2,  25,  26,   3,  23,  12,  34,
                 4,  13,  40,  32,   9,  13, 119,  10,  52,   7,  26,   8,   3,
                75,  24,  33,  31,  53,   7,   4,  89,   9,  25,   3,  68,  11,
                71,   4,  30,   5,  77,  47,  12,   3,  20,  34,  43,   7,  17,
                23,  13,   9,   1,  36, 101,   3,  10,   4,   3, 106,  40,  14,
                 8,  40,  30,   6,  20,  35,  31,   9,  40,  19,   6,  42,  10,
                61,  19,  68,  33, 288,  20,   2,   5,  15,  13,  50,   2,   3,
                 6,  15,  37,  70,   5,  29,  46,  18,  12,   3,  21,  72, 410,
                 9,  42,  16,   9,  27,  17,  13,  16,  38,  61,  11,  17,  40,
                 5,  33,   8,   5,  75,  15,  38,  19,  85,  31,  39,  20,   3,
                 8,  30,  29,  10,  71, 272,  80,  11,  35,  56,  64,   5,   6,
                17,   2,   7,  89, 359,  44,  17,   1,   5,   9,   2,  19,  29,
                10,   6,  33,  33,   6,  20,  34,   9,   6,  12,  56,   4,   6,
                34,   8,  14,  14,  12,  13,   4,  56,  14,  26,   4,  11,  17,
                34,   1,  17,   7,  62,   9,   6,  13,   6,  84,  34,  17,   8,
                71,  48,  14,   7,  44,  37,  56, 129,   9,  36,   5,  26,  20,
                56,  30,   9,  92,  51,  28, 141,  17,  29,  12,   5,  23,  64,
                42,  16, 126,  26,   3,  11,   3,  82, 256,  13,   6,   0,  72,
                75,  16,   7,  20,  21,  57,  17, 154,  75,   3,  11,   9,  20,
                18,  29,  56,  29,  33,  19,  21,  38,  38,  14,   2,  21,   3,
                12,  33, 191, 115, 395,  27,   0,  56,   6,  27,  11,   7,  28,
                28,  14,  30,  12,   8,  11,   8,  13,   1,   8,  76,  20,  14,
                31,  67,   6,  66,   7,  37,   9,  11,  14,  32,  19,   6,   2,
               102,   7,  26,   6,   3,   8,   3,  67,  48,  32,  31,  73,   5,
                18,  15,  59,  36,   6,  21,  10,  17,   7,  32,  25,  14,  18,
                 6,  60,  23,   4,   6,  24,  55,  26,   2,  11,  56,  22,  14,
                85,   5,  17,   4, 181,  40,  16,   7,  43,   3,  17, 142,  17,
                 9,   0,  25,  19,  24,  10, 129,  22,  21,  12,  13, 329,  47,
                48,  17,   4,  71,  16,  14,  46,  13,  90,   5,  27,   4,  12,
                18,  46, 112, 128,   7, 127,   4,  29,  16,   6,  34,  17,  70,
                31,   8,  13,  21,  25,  31,  34,   3,  41,  13,   9,  67,  22,
                18,  16,  23, 104,   7,   7, 202,  15,   4,   3,  14,  37,   6,
                15,  10,  35,  60,   0,  29,  15,  11,  16,  16,   3,  13,   2,
                21,  11,   7,  73,  15,  39,  19,  22,  20,  47,   8,  26,   6,
                45,  66,  10,  10,   3,   3,  41,  21,  22,  11,  10,  24,  55,
                 8,   3, 131,  68,   5, 102,  17,  14,  16,   5,  25,  64,  23,
                35,   9,   5,  42,  15,  10,  31,  21,  34,  23,   4,  32,  38,
                 4,   8,  27,   3,  37,  24,   4,  28,   3,   2,  76,  34,  18,
               127,   5,  13,  27,  30,  26, 302,  30,  92,  16,   4,  35,  30,
                 7,  49,   5,  53,  20,  47,  28,  13,  28,  13,  52,   4,   1],
              dtype=int64)
```

```
In [87]: lridge = Ridge(alpha=0.5, random_state=123)
In [88]: lridge.fit(X_train_scaled,y_train)

Out[88]: Ridge(alpha=0.5, copy_X=True, fit_intercept=True, max_iter=None,
               normalize=False, random_state=123, solver='auto', tol=0.001)
```

```
In [89]: ypredridge = lridge.predict(X_test_scaled)
```

```
In [90]: y_test[:10]

Out[90]: array([31, 26,  7,  5, 65,  3, 23,  1, 78,  4], dtype=int64)
```

```
In [91]: ypredridge[:10]

Out[91]: array([ 37.53806353,  15.61312738,  34.18539669,  38.36735229,
                 44.22282347, -11.3641019 ,   7.71916815,  -2.86776714,
                 65.20835978,  38.32572186])
```

Ridge Regression Model Evaluation

```
In [92]: mse = mean_squared_error(y_test,ypredridge)
         mse

Out[92]: 2265.173998629175
```

```
In [93]: rmse = np.sqrt(mse)
         rmse

Out[93]: 47.59384412536116
```

```
In [94]: r2_score(y_test,ypredridge)

Out[94]: 0.13105244491864254
```

Lasso Regression

```
In [95]: llasso = Lasso(alpha=0.5, random_state=123)
```

```
In [96]: llasso.fit(X_train_scaled,y_train)

Out[96]: Lasso(alpha=0.5, copy_X=True, fit_intercept=True, max_iter=1000,
               normalize=False, positive=False, precompute=False, random_state=123,
               selection='cyclic', tol=0.0001, warm_start=False)
```

```
In [97]: ypredlasso = llasso.predict(X_test_scaled)
```

```
In [98]: y_test[:10]

Out[98]: array([31, 26,  7,  5, 65,  3, 23,  1, 78,  4], dtype=int64)
```

```
In [99]: ypredlasso[:10]

Out[99]: array([41.40689525, 18.37196846, 23.46610452, 43.91767065, 45.51813448,
                -9.97304095,  8.67772412, -3.65533923, 59.60398402, 40.92403884])
```

Lasso Regression Model Evaluation

In [100]:

```
mse = mean_squared_error(y_test,ypredlasso)
mse
```

Out[100]:

```
2279.791001571065
```

In [101]:

```
rmse = np.sqrt(mse)
rmse
```

Out[101]:

```
47.74715699987869
```

In [102]:

```
r2_score(y_test,ypredlasso)
```

Out[102]:

```
0.1254451895922707
```

A paragraph explaining which of your regressions you recommend as a final model that best fits your needs in terms of accuracy and explainability.

Ridge Regression gives us the best score for lowest RMSE and highest R2 score.

Summary Key Findings and Insights, which walks your reader through the main drivers of your model and insights from your data derived from your linear regression model.

Based on the dataset provided, we found that Linear Regression model not suitable due to low R2 scores. Even with regularization, it only improves the model a bit.

We need to do extra research to find out what really causes dengue fever in San Jose. The environment and climate data is inadequate to explain dengue cases.

Suggestions for next steps in analyzing this data, which may include suggesting revisiting this model adding specific data features to achieve a better explanation or a better prediction.

We need to find out any other features beyond the current dataset provides as the prediction was not satisfactory.

We also need to explore other models like tree, support vector machine, random forest regressors model to see if they can able to analyse the data patterns to give better predictions.