Group 16
William (Billy) Edwards
Samihan Jawalkar
Shubham Shah
Reetesh Zope

Tasks

Through this notebook we will first clean the data to ensure any modeling can be performed on the data. This includes handling missing and null values, dropping outliers, and standardizing the data. After this part, we will go through exploratory data analysis to look for relationships amongst the variables. We will utilize box plots, scatter plot matrices, histogram matrices, and heat maps to look at these relationships. In particular, we will try to see if increasing humidity, precipitation, air temperature, and vegetation index is more likely to predict an increas in the the total Dengue cases for each (city, year, and weekofyear).

Dataset

The features in this dataset

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

(Where appropriate, units are provided as a _unit suffix on the feature name.) City and date indicators

```
city – City abbreviations: sj for San Juan and iq for Iquitos
week_start_date – Date given in yyyy-mm-dd format
```

NOAA's GHCN daily climate data weather station measurements

```
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
```

PERSIANN satellite precipitation measurements (0.25x0.25 degree scale)

precipitation_amt_mm – Total precipitation

NOAA's NCEP Climate Forecast System Reanalysis measurements (0.5x0.5 degree scale)

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

Satellite vegetation - Normalized difference vegetation index (NDVI) - NOAA's CDR Normalized Difference Vegetation Index (0.5x0.5 degree scale) measurements

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

Data Cleaning

```
#importing libraries
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
from scipy.stats import zscore
import scipy
import numpy as np
from scipy.cluster import hierarchy as sch


#reading the file
df = pd.read_csv('dengue_features_train.csv')
```

```
#looking through the data
df.head()
```

	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	preci
0	sj	1990	18	4/30/1990	0.122600	0.103725	0.198483	0.177617	
1	sj	1990	19	5/7/1990	0.169900	0.142175	0.162357	0.155486	
2	sj	1990	20	5/14/1990	0.032250	0.172967	0.157200	0.170843	
3	sj	1990	21	5/21/1990	0.128633	0.245067	0.227557	0.235886	
4	sj	1990	22	5/28/1990	0.196200	0.262200	0.251200	0.247340	

```
#descirbing the statistical information of data
df.describe()
```

	year	weekofyear	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	preci
count	1456.000000	1456.000000	1262.000000	1404.000000	1434.000000	1434.000000	
mean	2001.031593	26.503434	0.142294	0.130553	0.203783	0.202305	
std	5.408314	15.019437	0.140531	0.119999	0.073860	0.083903	
min	1990.000000	1.000000	-0.406250	-0.456100	-0.015533	-0.063457	
25%	1997.000000	13.750000	0.044950	0.049217	0.155087	0.144209	
50%	2002.000000	26.500000	0.128817	0.121429	0.196050	0.189450	
75%	2005.000000	39.250000	0.248483	0.216600	0.248846	0.246982	
max	2010.000000	53.000000	0.508357	0.454429	0.538314	0.546017	

Here we can get our first overview of our features. It is clear from the count values that several have missing data which we will later impute. It's also relevant to note the scales of some of the features are quite differnet such as the mean diurnal temperature range "reanalysis_tdtr_k" (4.9) versus the total precipitation amount "reanalysis_sat_precip_amt_mm" (45.76). These features will need to be standardized to avoid skewing predictions.

```
#getting info of all the columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1456 entries, 0 to 1455
Data columns (total 25 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   city                                   1456 non-null   object
```

```
 1   year                                   1456 non-null   int64
 2   weekofyear                             1456 non-null   int64
 3   week_start_date                        1456 non-null   object
 4   ndvi_ne                                1262 non-null   float64
 5   ndvi_nw                                1404 non-null   float64
 6   ndvi_se                                1434 non-null   float64
 7   ndvi_sw                                1434 non-null   float64
 8   precipitation_amt_mm                   1443 non-null   float64
 9   reanalysis_air_temp_k                  1446 non-null   float64
 10  reanalysis_avg_temp_k                  1446 non-null   float64
 11  reanalysis_dew_point_temp_k            1446 non-null   float64
 12  reanalysis_max_air_temp_k              1446 non-null   float64
 13  reanalysis_min_air_temp_k              1446 non-null   float64
 14  reanalysis_precip_amt_kg_per_m2        1446 non-null   float64
 15  reanalysis_relative_humidity_percent   1446 non-null   float64
 16  reanalysis_sat_precip_amt_mm           1443 non-null   float64
 17  reanalysis_specific_humidity_g_per_kg  1446 non-null   float64
 18  reanalysis_tdtr_k                      1446 non-null   float64
 19  station_avg_temp_c                     1413 non-null   float64
 20  station_diur_temp_rng_c                1413 non-null   float64
 21  station_max_temp_c                     1436 non-null   float64
 22  station_min_temp_c                     1442 non-null   float64
 23  station_precip_mm                      1434 non-null   float64
 24  total_cases                            1456 non-null   int64
dtypes: float64(20), int64(3), object(2)
memory usage: 284.5+ KB
```

From the output we see three different data types, of which only two are non-numerical. Both "city" and "week_start_date" are listed as objects. The city feature will be converted to a categorical variable, while "week_start_date" will be converted to date time. There are also 25 columns in the dataset and 1,456 rows of data.

```
#checking number of rows and columns
df.shape
```

```
(1456, 25)
```

```
#percentage of null values
df.isnull().sum() * 100 / len(df)
```

```
city                                     0.000000
year                                     0.000000
weekofyear                               0.000000
week_start_date                          0.000000
ndvi_ne                                 13.324176
ndvi_nw                                  3.571429
ndvi_se                                  1.510989
ndvi_sw                                  1.510989
precipitation_amt_mm                     0.892857
reanalysis_air_temp_k                    0.686813
reanalysis_avg_temp_k                    0.686813
```

```
reanalysis_dew_point_temp_k              0.686813
reanalysis_max_air_temp_k                0.686813
reanalysis_min_air_temp_k                0.686813
reanalysis_precip_amt_kg_per_m2          0.686813
reanalysis_relative_humidity_percent     0.686813
reanalysis_sat_precip_amt_mm             0.892857
reanalysis_specific_humidity_g_per_kg    0.686813
reanalysis_tdtr_k                        0.686813
station_avg_temp_c                       2.953297
station_diur_temp_rng_c                  2.953297
station_max_temp_c                       1.373626
station_min_temp_c                       0.961538
station_precip_mm                        1.510989
total_cases                              0.000000
dtype: float64
```

Listed above, we see that all but five of the variables have missing data. Since the data missing is under 50%, we will impute the values based on the median of the variable. The median was chosen since most of the variable have outliers and imputing on the mean would capture more of that skewness in the data.

```
#imputing median for null values
for i,colname in enumerate(list(df.columns)):
  if df[colname].isnull().sum() * 100 / len(df):
    df[colname].fillna(df[colname].median(), inplace=True)

df.isnull().sum() * 100 / len(df)
```

```
city                                     0.0
year                                     0.0
weekofyear                               0.0
week_start_date                          0.0
ndvi_ne                                  0.0
ndvi_nw                                  0.0
ndvi_se                                  0.0
ndvi_sw                                  0.0
precipitation_amt_mm                     0.0
reanalysis_air_temp_k                    0.0
reanalysis_avg_temp_k                    0.0
reanalysis_dew_point_temp_k              0.0
reanalysis_max_air_temp_k                0.0
reanalysis_min_air_temp_k                0.0
reanalysis_precip_amt_kg_per_m2          0.0
reanalysis_relative_humidity_percent     0.0
reanalysis_sat_precip_amt_mm             0.0
reanalysis_specific_humidity_g_per_kg    0.0
reanalysis_tdtr_k                        0.0
station_avg_temp_c                       0.0
station_diur_temp_rng_c                  0.0
station_max_temp_c                       0.0
station_min_temp_c                       0.0
station_precip_mm                        0.0
```

```
total_cases                             0.0
dtype: float64
```

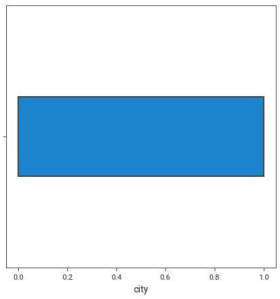
```
#converting city into categorical varible using label encoding
label_encoder = LabelEncoder()
df['city'] = label_encoder.fit_transform(df['city'])
```

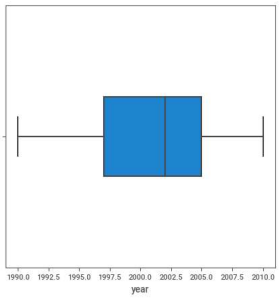
```
df['city']
```

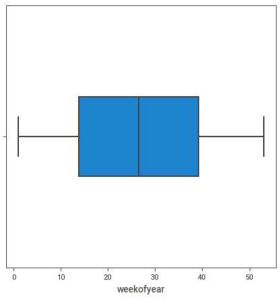
```
0       1
1       1
2       1
3       1
4       1
       ..
1451    0
1452    0
1453    0
1454    0
1455    0
Name: city, Length: 1456, dtype: int64
```

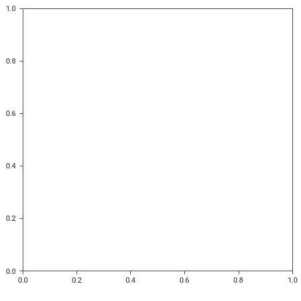
Exploratory Data Analysis

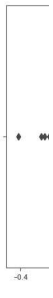
```
#boxplot matrix
fig, axes = plt.subplots(5,5, figsize=(30, 30)) # create figure and axes

for i,el in enumerate(list(df.columns)):
    if el == 'week_start_date':
      continue
    ax = axes.flatten()[i]
    sns.boxplot(x=el,data=df,width=0.3,ax=ax)

plt.show()
```

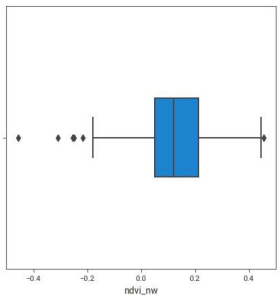

0.0 0.2 0.4 0.6 0.8 1.0
city

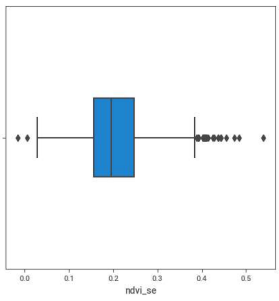

1990.0 1992.5 1995.0 1997.5 2000.0 2002.5 2005.0 2007.5 2010.0
year

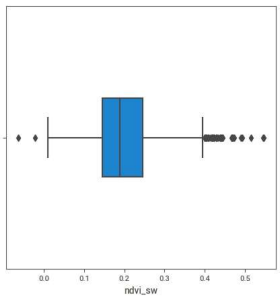

0 10 20 30 40 50
weekofyear

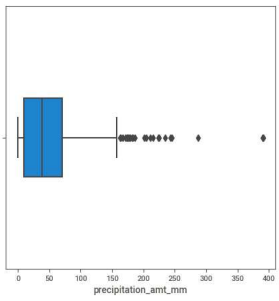

0.0 0.2 0.4 0.6 0.8 1.0


-0.4

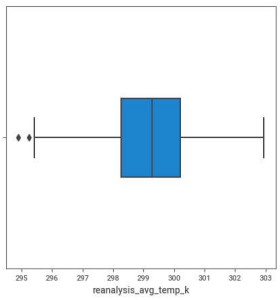

-0.4 -0.2 0.0 0.2 0.4
ndvi_nw

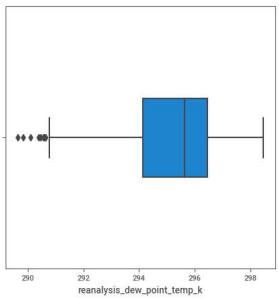

0.0 0.1 0.2 0.3 0.4 0.5
ndvi_se

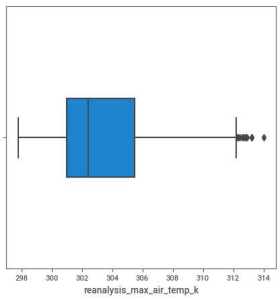

0.0 0.1 0.2 0.3 0.4 0.5
ndvi_sw

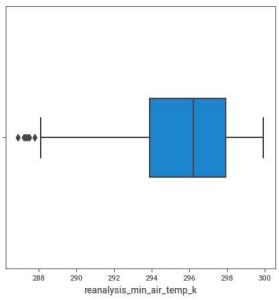

0 50 100 150 200 250 300 350 400
precipitation_amt_mm

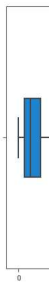

295

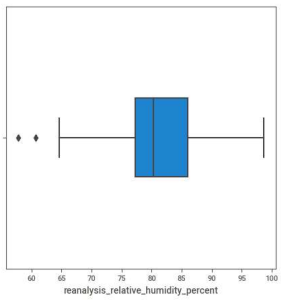

295 296 297 298 299 300 301 302 303
reanalysis_avg_temp_k

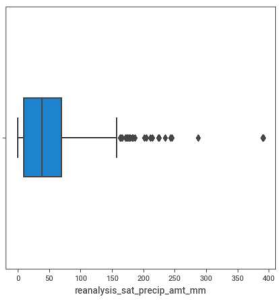

290 292 294 296 298
reanalysis_dew_point_temp_k

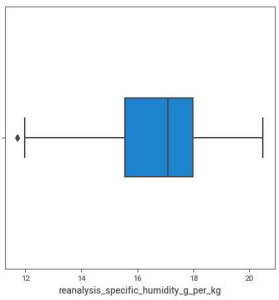

298 300 302 304 306 308 310 312 314
reanalysis_max_air_temp_k

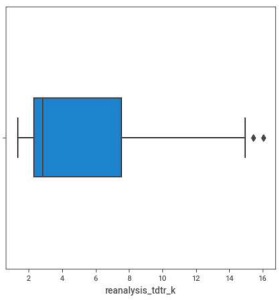

288 290 292 294 296 298 300
reanalysis_min_air_temp_k


0

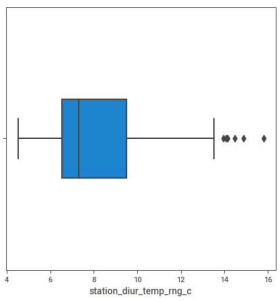

60 65 70 75 80 85 90 95 100
reanalysis_relative_humidity_percent

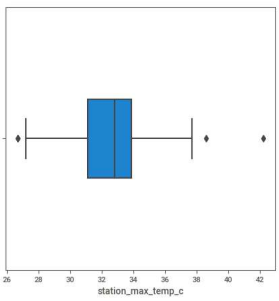

0 50 100 150 200 250 300 350 400
reanalysis_sat_precip_amt_mm

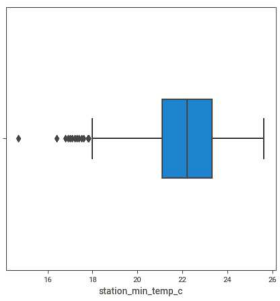

12 14 16 18 20
reanalysis_specific_humidity_g_per_kg

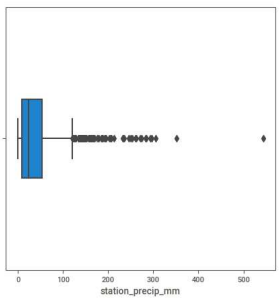

2 4 6 8 10 12 14 16
reanalysis_tdtr_k

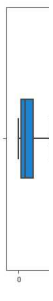

22


4 6 8 10 12 14 16
station_diur_temp_rng_c


26 28 30 32 34 36 38 40 42
station_max_temp_c


16 18 20 22 24 26
station_min_temp_c


0 100 200 300 400 500
station_precip_mm


0

From the above box plot matrix, we see that 20 out of the 25 box plots show outliers. "Total_cases" is the target variable and will not have any outliers dropped. Most of the plots show more than seven outliers, and these outliers are most likely not due to errors when recording the data. To avoid skewing the first linear regression prediciton we will try standardizing the variables with more than seven outliers. Those with less than seven outliers will be dropped and assumed that they were recorded by error. We will also incorporate other models such as random forest that will more easily handle outliers, and we can keep them in the data to see which model gives the best accuracy.

```
#calculting the z score, removing outliers that are more than 3 standard deviations from the
outlier_variables = ['ndvi_ne','ndvi_nw','reanalysis_avg_temp_k', 'reanalysis_max_air_temp_k'
,'reanalysis_min_air_temp_k','reanalysis_relative_humidity_percent','reanalysis_specific_humi
,'reanalysis_tdtr_k','station_avg_temp_c','station_diur_temp_rng_c','station_max_temp_c']
z_scores = zscore(df[outlier_variables])
abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
new_df = df[filtered_entries]
```

```
new_df.shape
```

```
(1423, 25)
```

```
#stad scaling
from sklearn.preprocessing import StandardScaler
trans_df = new_df
scaler = StandardScaler()
trans_df[outlier_variables] = scaler.fit_transform(new_df[outlier_variables])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: 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_
  """
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734: SettingWithCopyWarr
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_
  isetter(loc, value[:, i].tolist())
```

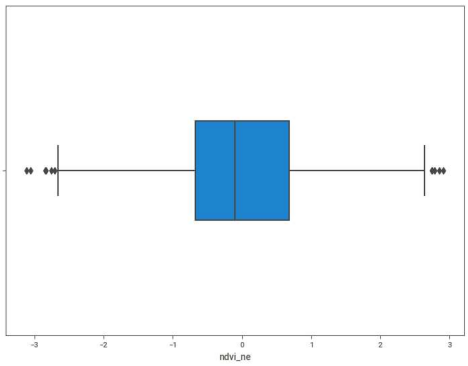
```
trans_df[outlier_variables]
```

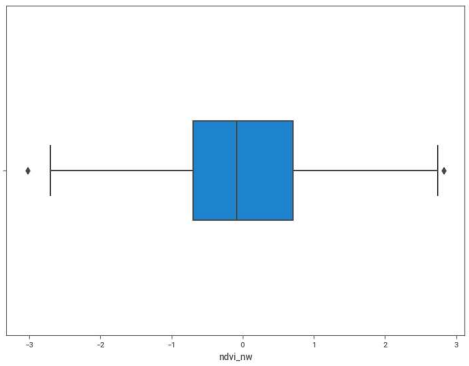
	ndvi_ne	ndvi_nw	reanalysis_avg_temp_k	reanalysis_max_air_temp_k	reanalysis_
0	-0.149258	-0.236646	-1.214027	-1.131498	
1	0.225219	0.099232	-0.644863	-0.783377	
2	-0.864563	0.368211	-0.290588	-0.909966	
3	-0.101492	0.998036	-0.006006	-0.625140	
4	0.433437	1.147703	0.348270	-0.466904	
...	...	...	...	...	
1451	1.593681	1.643002	1.248478	2.001588	
1452	0.148084	0.258185	0.127574	1.621820	
1453	0.836076	0.133144	-1.335991	0.672400	
1454	1.523728	1.004192	-1.394069	0.862284	
1455	1.240863	0.892378	0.487657	1.400289	

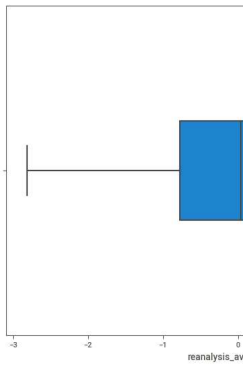
1423 rows × 11 columns

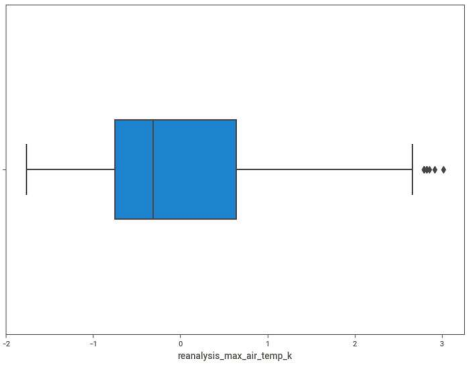
Above, we have standardized the variables that had outliers removed. Variables such as "reanalysis_max_air_temp_k" were on a much larger scale with a mean of (303.43) compared to that of the vegetation indices. Standardizing the variables now makes them more comparable and will make them less susecptible to skewing prediction results.

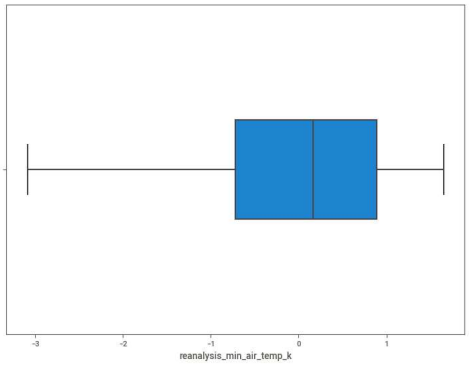
```
#boxplot after removing outlier
fig, axes = plt.subplots(4,3, figsize=(30, 30)) # create figure and axes

for i,el in enumerate(list(outlier_variables)):
    if el == 'week_start_date':
      continue
    ax = axes.flatten()[i]
    sns.boxplot(x=el,data=trans_df,width=0.3,ax=ax)

plt.show()
```

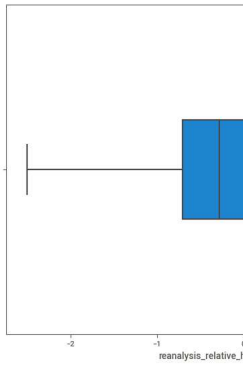

ndvi_ne

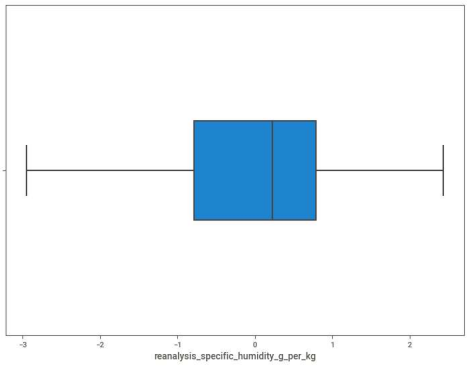

ndvi_nw

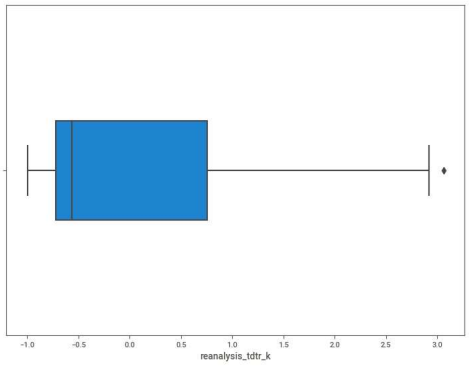


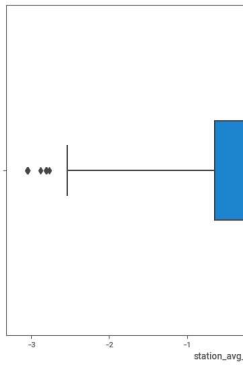

reanalysis_max_air_temp_k

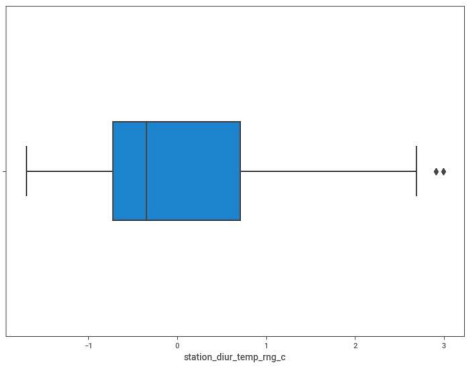

reanalysis_min_air_temp_k

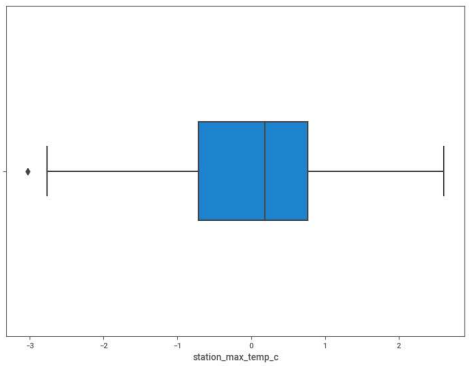


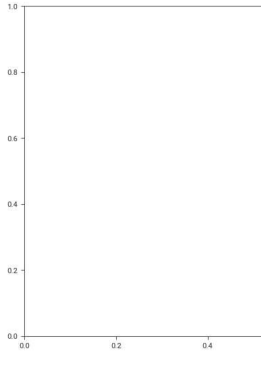

reanalysis_specific_humidity_g_per_kg


reanalysis_tdtr_k




station_diur_temp_rng_c


station_max_temp_c



We have plotted a new box plot matrix for the standardized variables. We can see that each of them no show less outliers than before, after they each had outliers that were 3 or more standard deviations away dropped. A few variables like "station_diur_temp_rng_c" and "station_max_temp_c" still show some outlier behavior. After running a regression these variables may need to be log transformed to handle the continued outlier influence on the prediction.

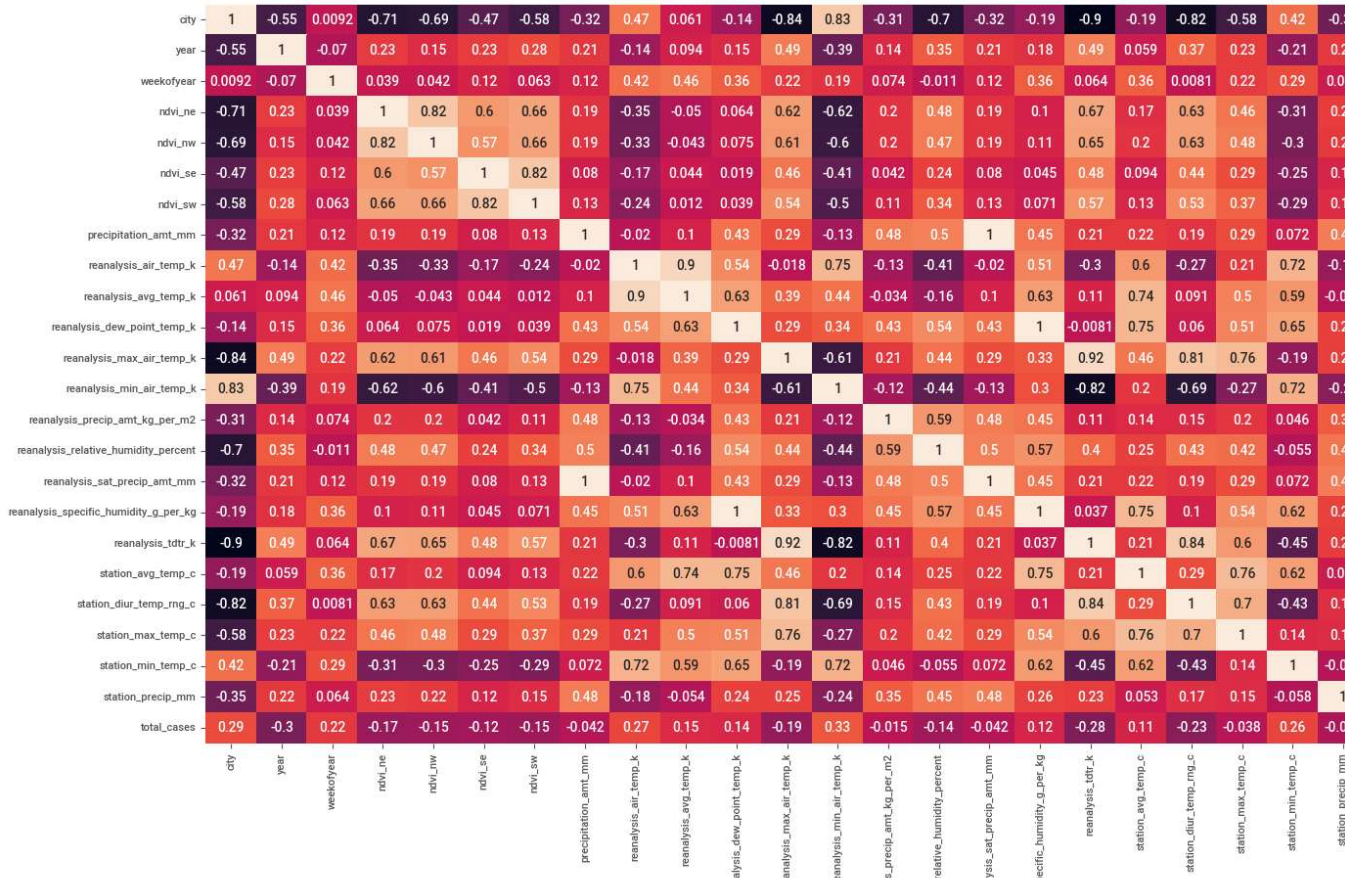
```
#getting correlation matrix
corr = trans_df.corr()
corr
```

	city	year	weekofyear	ndvi_ne	ndvi_nw
city	1.000000	-0.551265	0.009217	-0.709099	-0.688999
year	-0.551265	1.000000	-0.069893	0.226798	0.154095
weekofyear	0.009217	-0.069893	1.000000	0.038611	0.041848
ndvi_ne	-0.709099	0.226798	0.038611	1.000000	0.818041
ndvi_nw	-0.688999	0.154095	0.041848	0.818041	1.000000
ndvi_se	-0.470289	0.230816	0.115232	0.600005	0.566472
ndvi_sw	-0.576480	0.275410	0.063041	0.656272	0.657101
precipitation_amt_mm	-0.316401	0.211250	0.117197	0.189899	0.186434
reanalysis_air_temp_k	0.465149	-0.136574	0.419061	-0.345661	-0.331345
reanalysis_avg_temp_k	0.060869	0.094064	0.457263	-0.049711	-0.042635
reanalysis_dew_point_temp_k	-0.139302	0.148753	0.355199	0.064090	0.074696

A correlation table was created to look for multicollinearity amongst the variables.

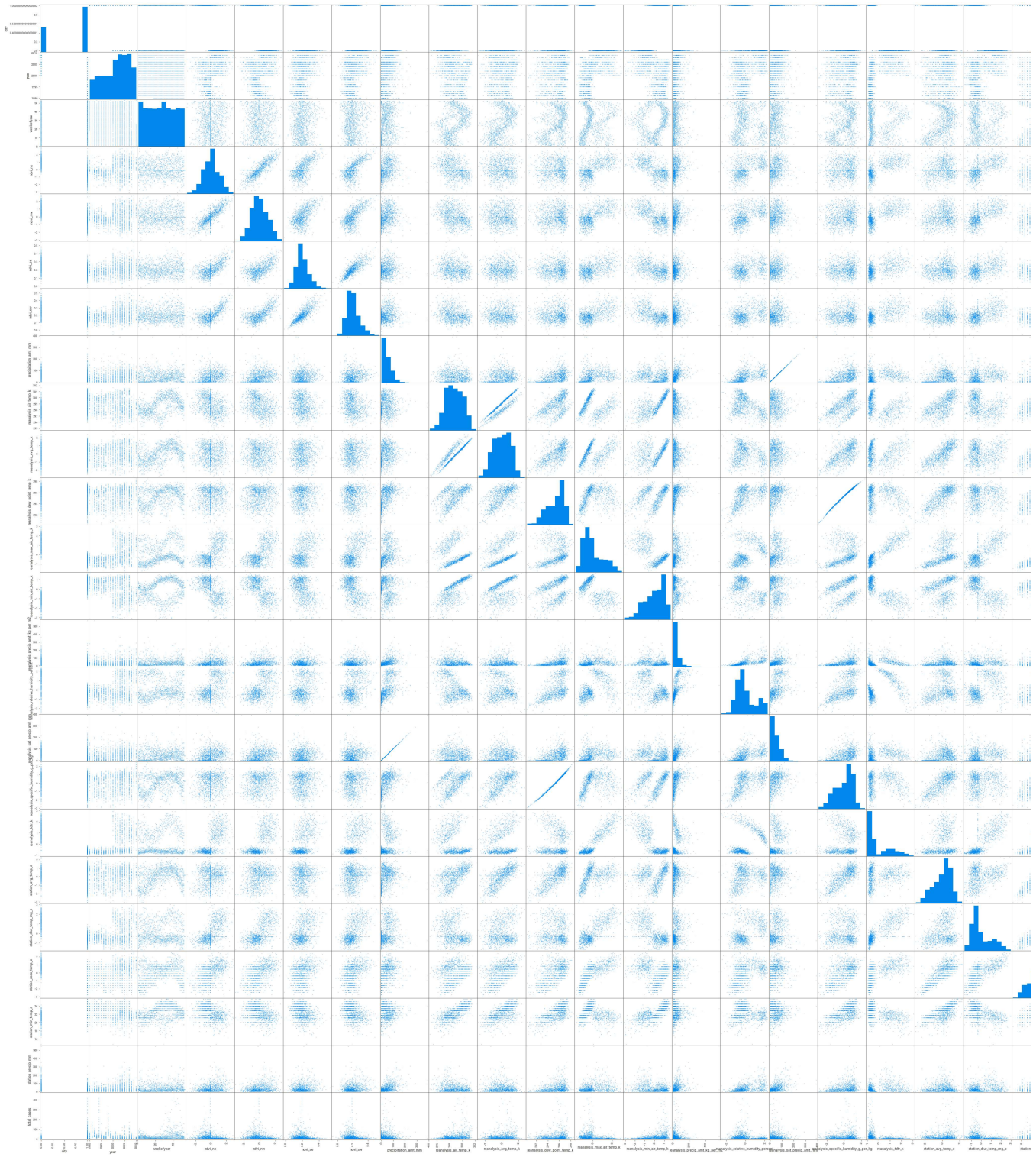
```
#plotting heat map
ax, fig = plt.subplots(figsize=[20,10])
sns.heatmap(corr, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe06be807d0>
```



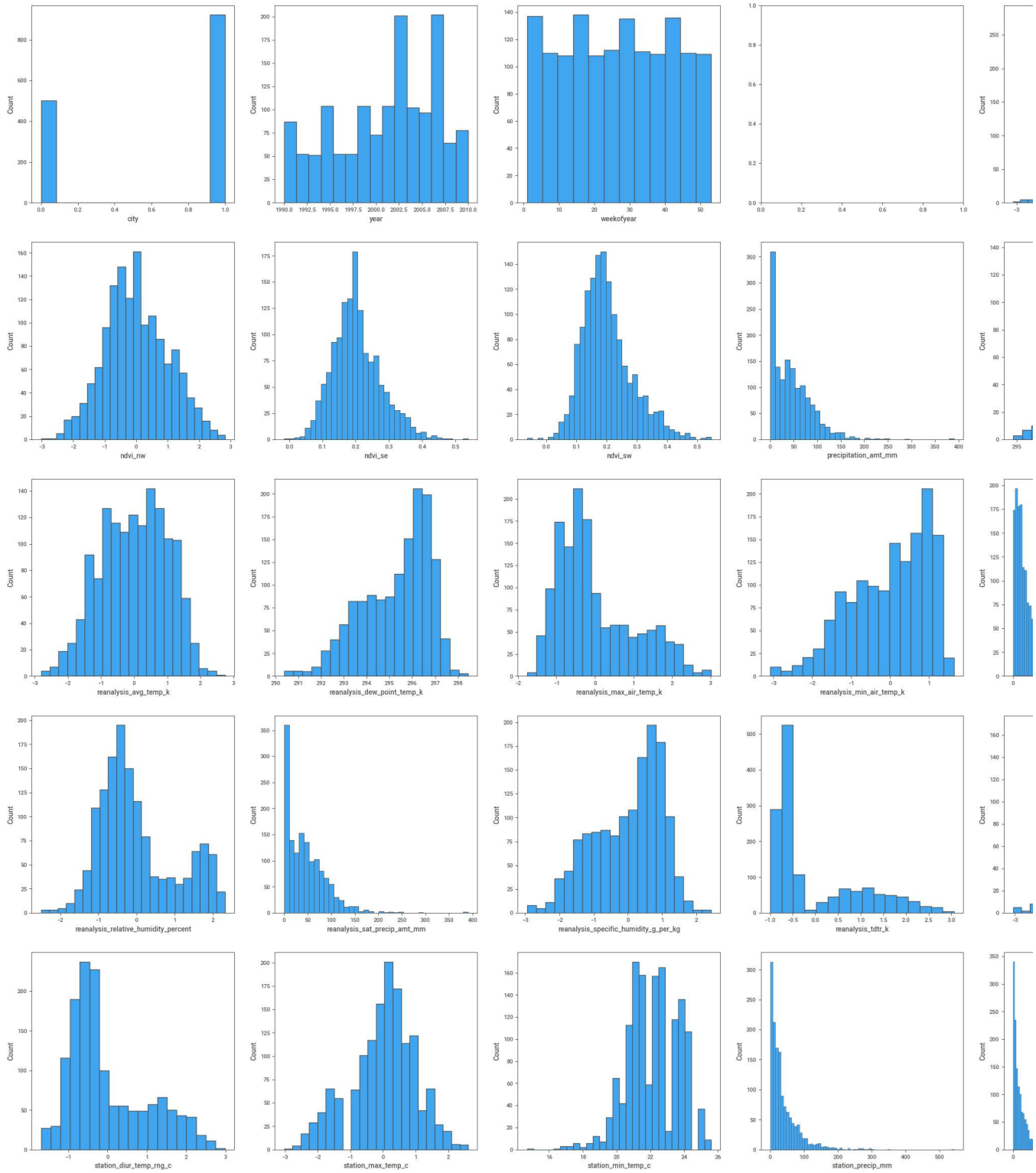
A heat map of the correlation matrix was generated. The lighter the color, the more positively correlated it is with a variable and vice-versa. This map makes it easier to spot multicollinearity. Here, there are many variables that might influence one another. For example, the satellite vegetation indices "ndvi" all seem to influence each other somewhat. The southeast centroid is 0.82 correlated with the southwest centroid. It may be best to keep only one southeast centroid. After running a regression and looking at VIF scores to confirm their influence on each other, one can be dropped from the model.

```
#plotting scatter matrix
scatter_matrix(trans_df,figsize=(60,60))
plt.show()
```



From the revised scatter plot matrix, we see several variables that are positively correlated with the total case count. These variables are: weekofyear, reanalysis_air_temp_k, reanalysis_avg_temp_k, reanalysis_dew_point_temp_k, reanalysis_min_air_temp_k, station_avg_temp_c, and station_min_temp_c. When holding each other variable equal, these variables will produce a larger amount of dengue cases. It is interesting to note that more total cases occur as it gets later in the year. This coincides with the rainy season in both cities. We can also see that typically as the air temperature, dew point, and humidity increase the more cases that arrise.

```
#plotting histogram for all variable to check skewness
fig, axes = plt.subplots(5,5, figsize=(30, 30))
for i,el in enumerate(list(trans_df.columns)):
    if el == 'week_start_date':
      continue
    ax = axes.flatten()[i]

    sns.histplot(trans_df[el] ,ax=axes.flatten()[i])

plt.show()
```


Count
city
year
weekofyear
ndvi_nw
ndvi_se
ndvi_sw
precipitation_amt_mm
reanalysis_avg_temp_k
reanalysis_dew_point_temp_k
reanalysis_max_air_temp_k
reanalysis_min_air_temp_k
reanalysis_relative_humidity_percent
reanalysis_sat_precip_amt_mm
reanalysis_specific_humidity_g_per_kg
reanalysis_tdtr_k
station_diur_temp_rng_c
station_max_temp_c
station_min_temp_c
station_precip_mm

We have plotted a histogram matrix above for all the numerical variables. This is after dropping outliers and standardizing those variables that had outliers dropped. There is still some left and right tail skewness ocurring in the distributions. For example, "reanalyis_dew_point_temp_k" is left tail skewed. This variable will most likly cause increasing case predictions if it is left as is. After running a regression, it can be transformed to account for the skewness. "Station_precip_mm" works in the opposite way as it is right tail skewed. It may also need to be transformed after running a regression and looking at the output coefficients.

```
# Create a mask for values above 90%
# But also below 100% since it variables correlated with the same one
mask = (trans_df.corr() > 0.9) & (trans_df.corr() < 1.0)
high_corr_df = corr[mask]
```

```
#any will return item if itterable are true
col_to_filter_out = ~high_corr_df[mask].any()
```

```
#items which are false has high correlation,
col_to_filter_out
```

```
city                                    True
year                                    True
weekofyear                              True
ndvi_ne                                 True
ndvi_nw                                 True
ndvi_se                                 True
ndvi_sw                                 True
precipitation_amt_mm                    True
reanalysis_air_temp_k                   True
reanalysis_avg_temp_k                   True
reanalysis_dew_point_temp_k            False
reanalysis_max_air_temp_k              False
reanalysis_min_air_temp_k               True
reanalysis_precip_amt_kg_per_m2         True
reanalysis_relative_humidity_percent    True
reanalysis_sat_precip_amt_mm            True
reanalysis_specific_humidity_g_per_kg  False
reanalysis_tdtr_k                      False
station_avg_temp_c                      True
station_diur_temp_rng_c                 True
station_max_temp_c                      True
station_min_temp_c                      True
station_precip_mm                       True
total_cases                             True
dtype: bool
```

A mask variable was created to find variables in the correlation matrix that would be the most correlated and interfere with prediction results. All boolean values for the variables are shown. Anything that has a correlation below 90% is listed as True and anything with above 90% is listed as False. The False columns can be dropped as they will most likely to be multicollinear.

```
X_clean = df[high_corr_df.columns[col_to_filter_out]]
```

```
X_clean
```

	city	year	weekofyear	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_m
0	1	1990	18	0.122600	0.103725	0.198483	0.177617	12.4
1	1	1990	19	0.169900	0.142175	0.162357	0.155486	22.8
2	1	1990	20	0.032250	0.172967	0.157200	0.170843	34.5
3	1	1990	21	0.128633	0.245067	0.227557	0.235886	15.3
4	1	1990	22	0.196200	0.262200	0.251200	0.247340	7.5
...	...	...	...	...	...	...	...	.
1451	0	2010	21	0.342750	0.318900	0.256343	0.292514	55.3
1452	0	2010	22	0.160157	0.160371	0.136043	0.225657	86.4
1453	0	2010	23	0.247057	0.146057	0.250357	0.233714	58.9
1454	0	2010	24	0.333914	0.245771	0.278886	0.325486	59.6
1455	0	2010	25	0.298186	0.232971	0.274214	0.315757	63.2

1456 rows × 20 columns

New clean dataframe with variables that did not have correlation above 90% with another variable.

```
X_clean.shape
```

```
(1456, 20)
```

```
!pip install sweetviz
```

```
Requirement already satisfied: sweetviz in /usr/local/lib/python3.7/dist-packages (2.1.3
Requirement already satisfied: jinja2>=2.11.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: importlib-resources>=1.2.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: tqdm>=4.43.0 in /usr/local/lib/python3.7/dist-packages (4
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: matplotlib>=3.1.3 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.7/dist-packages (4
```

```
Requirement already satisfied: pandas!=1.0.0,!=1.0.1,!=1.0.2,>=0.25.3 in /usr/local/lib/
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/li
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycle
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (f
```

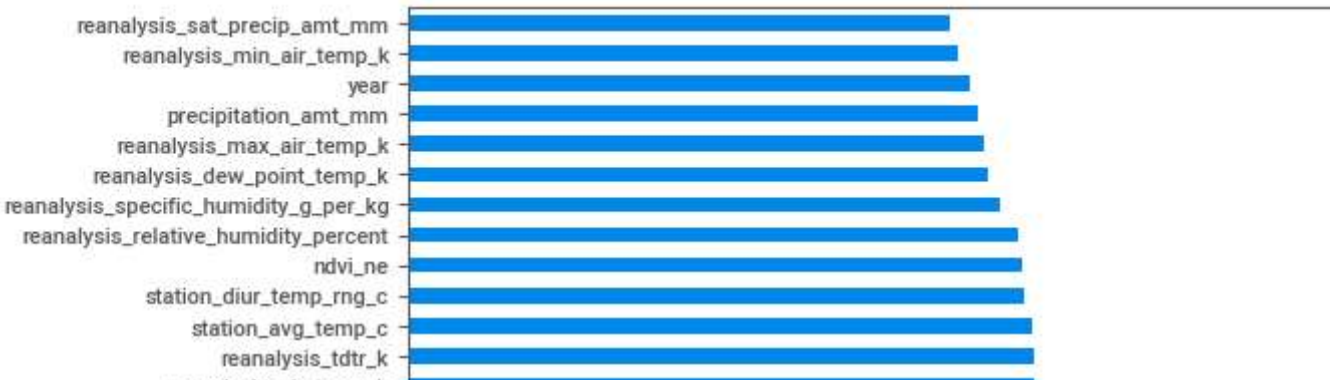
```
#getting report consisting association graph
import sweetviz as sv
my_report = sv.analyze(trans_df)
my_report.show_html()
```

```
Done! Use 'show' commands to display/save.                                    [100%]

Report SWEETVIZ_REPORT.html was generated! NOTEBOOK/COLAB USERS: the web browser MAY not
```

From the sweetviz page we can view the association plot. The squares are listed to depict how much one variable can imply another variable will occur. For example, the "city" variable will provide the vegetation index "ndvi" and the direction variables for it. A blue value is a positive implication, while a red value is negative. The circles show correlation coefficients like the heat map above. This gives another overview of multicollinearity that might occur when making predictions.

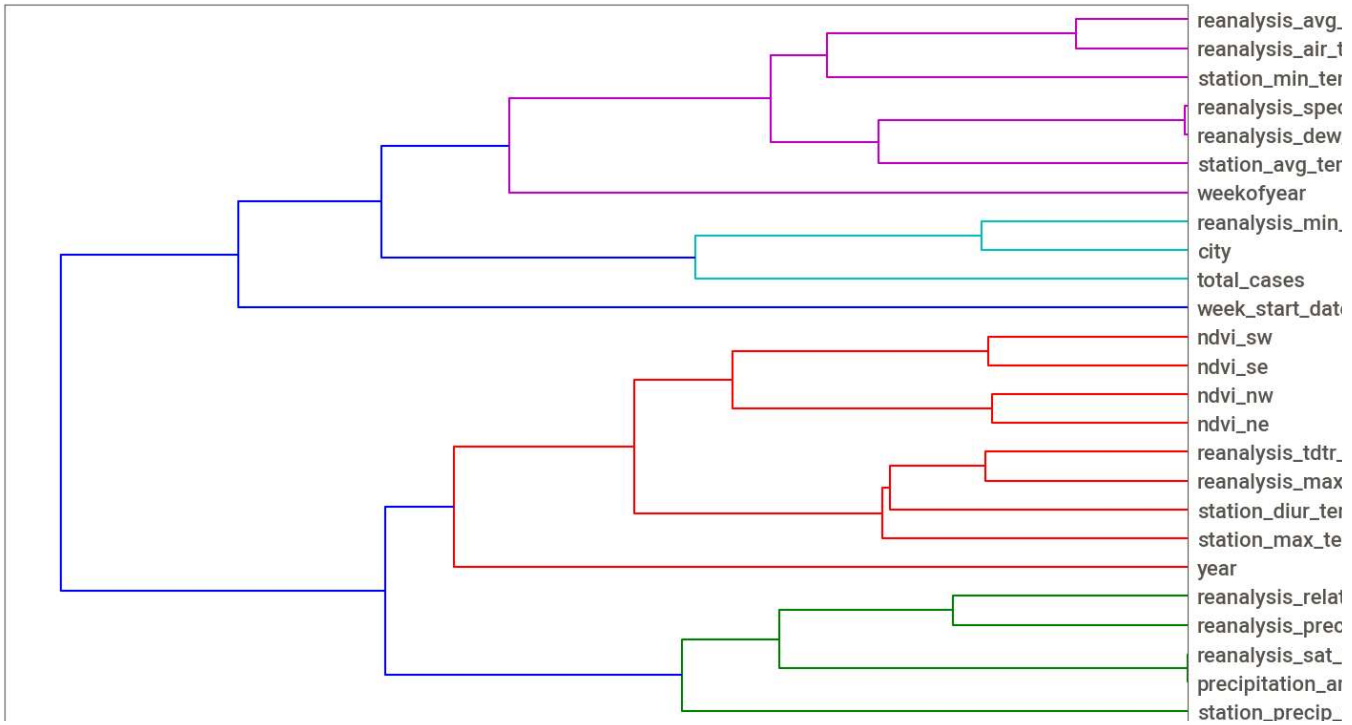
```
#getting the feature importance,
#using random forest model to know which feature has greater impact on dependent variable
from sklearn.ensemble import RandomForestClassifier
trans_df.dropna(axis=0,how='any',inplace=True)
X = trans_df.drop(["total_cases","week_start_date"] , axis=1)
#X['DAY_DELTA'] = (X.TIMESTAMP - X.TIMESTAMP.min()).dt.days

y = trans_df['week_start_date']
model = RandomForestClassifier()
model.fit(X, y)

(pd.Series(model.feature_importances_, index=X.columns)
   .nlargest(20)
   .plot(kind='barh'))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  after removing the cwd from sys.path.
<matplotlib.axes._subplots.AxesSubplot at 0x7fe078a194d0>
```



We have run a random forest model above which lists the feature importance for each remaining varaible in the data set in descending order. "Weekofyear" is shown as the best overall feature. When running our regression prediction, we can use this to create a threshold value. Any feature with less than 0.045 importance can be thrown out, and the remaining features can be kept. In this case, starting with "reanalysis_specific_humidity_g_per_kg" and below can be dropped before runnning a regression.

```
#correlation and heirarchical clustering using dendogram for all variable
corr = np.round(scipy.stats.spearmanr(trans_df).correlation, 4)
corr_condensed = sch.distance.squareform(1-corr)
z = sch.linkage(corr_condensed, method='average')
fig = plt.figure(figsize=(16,10))
dendrogram = sch.dendrogram(z, labels=trans_df.columns, orientation='left', leaf_font_size=16
plt.show()
```



The dendrogram above gives a visual on each feature, and depicts clustering relationship based on how related they are to other features. Again, we can view the correlation of the variables and see that variable such as "ndvi" and its directions are all quite correlated with two clusters next to each other. After running a prediction model, only one may need to be kept to best model a case of dengue.

```
#correlation and heirarchical clustering using dendogram for variables after dropping those t
corr = np.round(scipy.stats.spearmanr(X_clean).correlation, 4)
corr_condensed = sch.distance.squareform(1-corr)
z = sch.linkage(corr_condensed, method='average')
fig = plt.figure(figsize=(16,10))
dendrogram = sch.dendrogram(z, labels=X_clean.columns, orientation='left', leaf_font_size=16)
plt.show()
```