analysis

June 20, 2021

[1]:
```
%load_ext autoreload
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

1 Loading the Train Data

[2]:
```
train_data = pd.read_csv('data/dengue_features_train.csv')
train_data['total_cases'] = pd.read_csv('data/dengue_labels_train.
 ↪csv')['total_cases']
print(train_data.shape)
train_data.head()
```

```
(1456, 25)
```

[2]:

	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	\
0	sj	1990	18	1990-04-30	0.122600	0.103725	0.198483	
1	sj	1990	19	1990-05-07	0.169900	0.142175	0.162357	
2	sj	1990	20	1990-05-14	0.032250	0.172967	0.157200	
3	sj	1990	21	1990-05-21	0.128633	0.245067	0.227557	
4	sj	1990	22	1990-05-28	0.196200	0.262200	0.251200	

	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	…	\
0	0.177617	12.42	297.572857	…	
1	0.155486	22.82	298.211429	…	
2	0.170843	34.54	298.781429	…	
3	0.235886	15.36	298.987143	…	
4	0.247340	7.52	299.518571	…	

	reanalysis_relative_humidity_percent	reanalysis_sat_precip_amt_mm	\
0	73.365714	12.42	
1	77.368571	22.82	
2	82.052857	34.54	
3	80.337143	15.36	
4	80.460000	7.52	

	reanalysis_specific_humidity_g_per_kg	reanalysis_tdtr_k	\

```
0                              14.012857           2.628571
1                              15.372857           2.371429
2                              16.848571           2.300000
3                              16.672857           2.428571
4                              17.210000           3.014286

   station_avg_temp_c  station_diur_temp_rng_c  station_max_temp_c  \
0           25.442857                 6.900000                29.4
1           26.714286                 6.371429                31.7
2           26.714286                 6.485714                32.2
3           27.471429                 6.771429                33.3
4           28.942857                 9.371429                35.0

   station_min_temp_c  station_precip_mm  total_cases
0                20.0               16.0            4
1                22.2                8.6            5
2                22.8               41.4            4
3                23.3                4.0            3
4                23.9                5.8            6

[5 rows x 25 columns]
```

[3]:
```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1456 entries, 0 to 1455
Data columns (total 25 columns):
city                                   1456 non-null object
year                                   1456 non-null int64
weekofyear                             1456 non-null int64
week_start_date                        1456 non-null object
ndvi_ne                                1262 non-null float64
ndvi_nw                                1404 non-null float64
ndvi_se                                1434 non-null float64
ndvi_sw                                1434 non-null float64
precipitation_amt_mm                   1443 non-null float64
reanalysis_air_temp_k                  1446 non-null float64
reanalysis_avg_temp_k                  1446 non-null float64
reanalysis_dew_point_temp_k            1446 non-null float64
reanalysis_max_air_temp_k              1446 non-null float64
reanalysis_min_air_temp_k              1446 non-null float64
reanalysis_precip_amt_kg_per_m2        1446 non-null float64
reanalysis_relative_humidity_percent   1446 non-null float64
reanalysis_sat_precip_amt_mm           1443 non-null float64
reanalysis_specific_humidity_g_per_kg  1446 non-null float64
reanalysis_tdtr_k                      1446 non-null float64
station_avg_temp_c                     1413 non-null float64
station_diur_temp_rng_c                1413 non-null float64
```

```
station_max_temp_c                       1436 non-null float64
station_min_temp_c                       1442 non-null float64
station_precip_mm                        1434 non-null float64
total_cases                              1456 non-null int64
dtypes: float64(20), int64(3), object(2)
memory usage: 284.5+ KB
```

1.1 Nulls per attribute (percentage)

[4]:
```
train_data.isnull().mean() * 100
```

[4]:
```
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
```

1.2 Noise

- While plotting the data, we realized there are a few entries in the dataset that must be immediatly remove. They report a 53rd week of the year and have all values at *NaN*
- Also, there are some week 52's on the 1st month of some years. But these don't have all values at *NaN*, so perhaps we should retain them. Changing those weeks to week 0 is perhaps the most logical solution.

[5]:
```
train_data[train_data['weekofyear'] == 53]
```

[5]:
```
     city  year  weekofyear week_start_date  ndvi_ne  ndvi_nw  ndvi_se  \
139    sj  1993          53      1993-01-01      NaN      NaN      NaN
451    sj  1999          53      1999-01-01      NaN      NaN      NaN
763    sj  2005          53      2005-01-01      NaN      NaN      NaN
1170   iq  2005          53      2005-01-01      NaN      NaN      NaN
1430   iq  2010          53      2010-01-01      NaN      NaN      NaN

      ndvi_sw  precipitation_amt_mm  reanalysis_air_temp_k  …  \
139       NaN                   NaN                    NaN  …
451       NaN                   NaN                    NaN  …
763       NaN                   NaN                    NaN  …
1170      NaN                   NaN                    NaN  …
1430      NaN                   NaN                    NaN  …

      reanalysis_relative_humidity_percent  reanalysis_sat_precip_amt_mm  \
139                                    NaN                           NaN
451                                    NaN                           NaN
763                                    NaN                           NaN
1170                                   NaN                           NaN
1430                                   NaN                           NaN

      reanalysis_specific_humidity_g_per_kg  reanalysis_tdtr_k  \
139                                     NaN                NaN
451                                     NaN                NaN
763                                     NaN                NaN
1170                                    NaN                NaN
1430                                    NaN                NaN

      station_avg_temp_c  station_diur_temp_rng_c  station_max_temp_c  \
139                  NaN                      NaN                 NaN
451                  NaN                      NaN                 NaN
763                  NaN                      NaN                 NaN
1170                 NaN                      NaN                 NaN
1430                 NaN                      NaN                 NaN

      station_min_temp_c  station_precip_mm  total_cases
139                  NaN                NaN           30
451                  NaN                NaN           59
763                  NaN                NaN           10
1170                 NaN                NaN            9
1430                 NaN                NaN            0

[5 rows x 25 columns]
```

[56]:
```
train_data = train_data[train_data['weekofyear'] != 53]
train_data.reset_index(drop=True, inplace=True)
```

[7]:
```
months = pd.Series([x[5:7] for x in train_data['week_start_date']])
cond = np.logical_and((train_data['weekofyear'] == 52).tolist(), (months ==␣
↪'01').tolist())
train_data[cond]
```

[7]:

	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	\
191	sj	1994	52	1994-01-01	NaN	NaN	NaN	
243	sj	1995	52	1995-01-01	0.2438	0.045000	0.215256	
503	sj	2000	52	2000-01-01	-0.1336	-0.061225	0.168200	
815	sj	2006	52	2006-01-01	-0.1062	-0.137600	0.183500	
1222	iq	2006	52	2006-01-01	0.3546	0.219950	0.322517	

	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	…	\
191	NaN	5.25	298.527143	…	
243	0.194778	71.43	298.838571	…	
503	0.086311	40.35	298.367143	…	
815	0.169767	64.08	299.147143	…	
1222	0.256333	114.58	298.181429	…	

	reanalysis_relative_humidity_percent	reanalysis_sat_precip_amt_mm	\
191	78.018571	5.25	
243	82.867143	71.43	
503	76.317143	40.35	
815	76.578571	64.08	
1222	93.821429	114.58	

	reanalysis_specific_humidity_g_per_kg	reanalysis_tdtr_k	\
191	15.780000	2.442857	
243	17.067143	3.271429	
503	15.257143	2.642857	
815	16.027143	2.242857	
1222	18.584286	7.385714	

	station_avg_temp_c	station_diur_temp_rng_c	station_max_temp_c	\
191	26.971429	8.200000	32.8	
243	27.314286	8.742857	32.8	
503	25.471429	6.557143	29.4	
815	25.885714	6.271429	30.0	
1222	27.450000	9.700000	32.7	

	station_min_temp_c	station_precip_mm	total_cases
191	21.7	0.3	22
243	22.2	9.2	154
503	21.1	7.9	17
815	21.1	18.6	13
1222	22.4	273.5	8

[5 rows x 25 columns]

[8]:
```
train_data.loc[cond, 'weekofyear'] = 0
train_data[cond]
```

[8]:

	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	\
191	sj	1994	0	1994-01-01	NaN	NaN	NaN	
243	sj	1995	0	1995-01-01	0.2438	0.045000	0.215256	
503	sj	2000	0	2000-01-01	-0.1336	-0.061225	0.168200	
815	sj	2006	0	2006-01-01	-0.1062	-0.137600	0.183500	
1222	iq	2006	0	2006-01-01	0.3546	0.219950	0.322517	

	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	…	\
191	NaN	5.25	298.527143	…	
243	0.194778	71.43	298.838571	…	
503	0.086311	40.35	298.367143	…	
815	0.169767	64.08	299.147143	…	
1222	0.256333	114.58	298.181429	…	

	reanalysis_relative_humidity_percent	reanalysis_sat_precip_amt_mm	\
191	78.018571	5.25	
243	82.867143	71.43	
503	76.317143	40.35	
815	76.578571	64.08	
1222	93.821429	114.58	

	reanalysis_specific_humidity_g_per_kg	reanalysis_tdtr_k	\
191	15.780000	2.442857	
243	17.067143	3.271429	
503	15.257143	2.642857	
815	16.027143	2.242857	
1222	18.584286	7.385714	

	station_avg_temp_c	station_diur_temp_rng_c	station_max_temp_c	\
191	26.971429	8.200000	32.8	
243	27.314286	8.742857	32.8	
503	25.471429	6.557143	29.4	
815	25.885714	6.271429	30.0	
1222	27.450000	9.700000	32.7	

	station_min_temp_c	station_precip_mm	total_cases
191	21.7	0.3	22
243	22.2	9.2	154
503	21.1	7.9	17
815	21.1	18.6	13
1222	22.4	273.5	8

[5 rows x 25 columns]

2 Loading the Test Data

- The test data also contains *NaN* and noise, in a similar way to the train data.

[9]:
```
test_data = pd.read_csv('data/dengue_features_test.csv')
print(test_data.shape)
test_data.head()
```

(416, 24)

[9]:

	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	…
0	sj	2008	18	2008-04-29	-0.0189	-0.018900	0.102729	0.091200	78.60	298.492857	…
1	sj	2008	19	2008-05-06	-0.0180	-0.012400	0.082043	0.072314	12.56	298.475714	…
2	sj	2008	20	2008-05-13	-0.0015	NaN	0.151083	0.091529	3.66	299.455714	…
3	sj	2008	21	2008-05-20	NaN	-0.019867	0.124329	0.125686	0.00	299.690000	…
4	sj	2008	22	2008-05-27	0.0568	0.039833	0.062267	0.075914	0.76	299.780000	…

	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
0	25.37	78.781429	78.60	15.918571	3.128571	26.528571	7.057143
1	21.83	78.230000	12.56	15.791429	2.571429	26.071429	5.557143
2	4.12	78.270000	3.66	16.674286	4.428571	27.928571	7.785714
3	2.20	73.015714	0.00	15.775714	4.342857	28.057143	6.271429
4	4.36	74.084286	0.76	16.137143	3.542857	27.614286	7.085714

```
   station_max_temp_c  station_min_temp_c  station_precip_mm
0                33.3                21.7               75.2
1                30.0                22.2               34.3
2                32.8                22.8                3.0
3                33.3                24.4                0.3
4                33.3                23.3               84.1

[5 rows x 24 columns]
```

2.1 Nulls per attribute (percentage)

[10]:
```
test_data.isnull().mean() * 100
```

[10]:
```
city                                    0.000000
year                                    0.000000
weekofyear                              0.000000
week_start_date                         0.000000
ndvi_ne                                10.336538
ndvi_nw                                 2.644231
ndvi_se                                 0.240385
ndvi_sw                                 0.240385
precipitation_amt_mm                    0.480769
reanalysis_air_temp_k                   0.480769
reanalysis_avg_temp_k                   0.480769
reanalysis_dew_point_temp_k             0.480769
reanalysis_max_air_temp_k               0.480769
reanalysis_min_air_temp_k               0.480769
reanalysis_precip_amt_kg_per_m2         0.480769
reanalysis_relative_humidity_percent    0.480769
reanalysis_sat_precip_amt_mm            0.480769
reanalysis_specific_humidity_g_per_kg   0.480769
reanalysis_tdtr_k                       0.480769
station_avg_temp_c                      2.884615
station_diur_temp_rng_c                 2.884615
station_max_temp_c                      0.721154
station_min_temp_c                      2.163462
station_precip_mm                       1.201923
dtype: float64
```

[11]:
```
test_data[test_data['weekofyear'] == 53]
```

[11]:
```
    city  year  weekofyear week_start_date  ndvi_ne  ndvi_nw  ndvi_se  ndvi_sw  \
87    sj  2010          53      2010-01-01      NaN      NaN      NaN      NaN

    precipitation_amt_mm  reanalysis_air_temp_k  …  \
87                   NaN                    NaN  …
```

```
    reanalysis_precip_amt_kg_per_m2  reanalysis_relative_humidity_percent  \
87                               NaN                                   NaN

    reanalysis_sat_precip_amt_mm  reanalysis_specific_humidity_g_per_kg  \
87                           NaN                                   NaN

    reanalysis_tdtr_k  station_avg_temp_c  station_diur_temp_rng_c  \
87                NaN                 NaN                      NaN

    station_max_temp_c  station_min_temp_c  station_precip_mm
87                 NaN                 NaN                NaN

[1 rows x 24 columns]
```

[12]:
```
months = pd.Series([x[5:7] for x in test_data['week_start_date']])
cond = np.logical_and((test_data['weekofyear'] == 52).tolist(), (months ==␣
↪'01').tolist())
test_data[cond]
```

[12]:

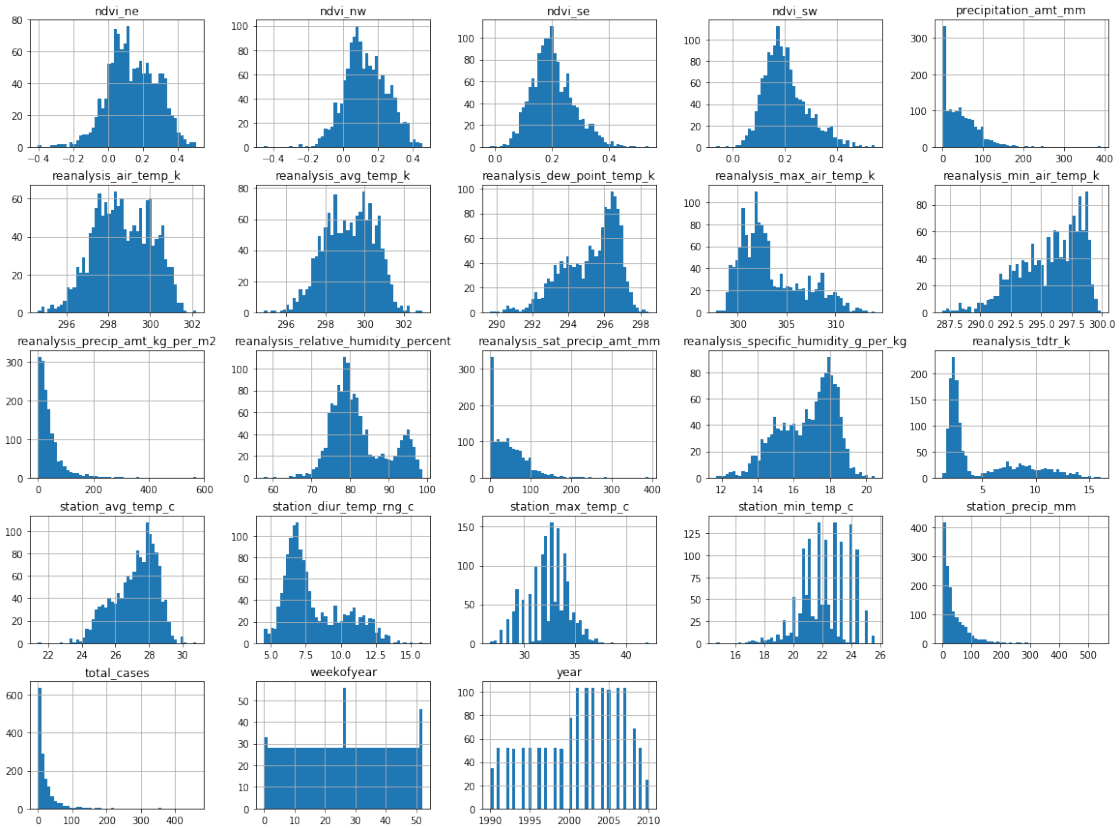
	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	…	reanalysis_precip_amt_kg_per_m2	reanalysis_relative_humidity_percent	reanalysis_sat_precip_amt_mm	reanalysis_specific_humidity_g_per_kg	reanalysis_tdtr_k	station_avg_temp_c	station_diur_temp_rng_c
139	sj	2011	52	2011-01-01	0.0650	NaN	0.162329	0.149911	7.57	298.247143	…	8.3	74.568571	7.57	14.717143	1.971429	25.428571	5.085714
191	sj	2012	52	2012-01-01	0.1653	0.099025	0.152875	0.081513	15.30	298.507143	…	8.0	73.187143	15.30	14.714286	2.042857	25.828571	5.785714
286	iq	2011	52	2011-01-01	0.2775	0.304367	0.235656	0.289433	43.67	296.571429	…	222.0	96.627143	43.67	17.517143	4.800000	26.900000	8.600000
338	iq	2012	52	2012-01-01	0.1926	0.241100	0.224662	0.184850	83.53	298.224286	…	120.4	93.264286	83.53	18.581429			

```
338          8.171429          28.500000          11.000000

     station_max_temp_c  station_min_temp_c  station_precip_mm
139                28.3                22.2                5.7
191                29.4                22.2               48.1
286                33.2                22.0               54.7
338                34.0                22.2               57.2

[4 rows x 24 columns]
```

3 Histograms

- None of the values happear to have been capped

[13]:
```
train_data.hist(bins=50, figsize=(20,15))
plt.show()
```

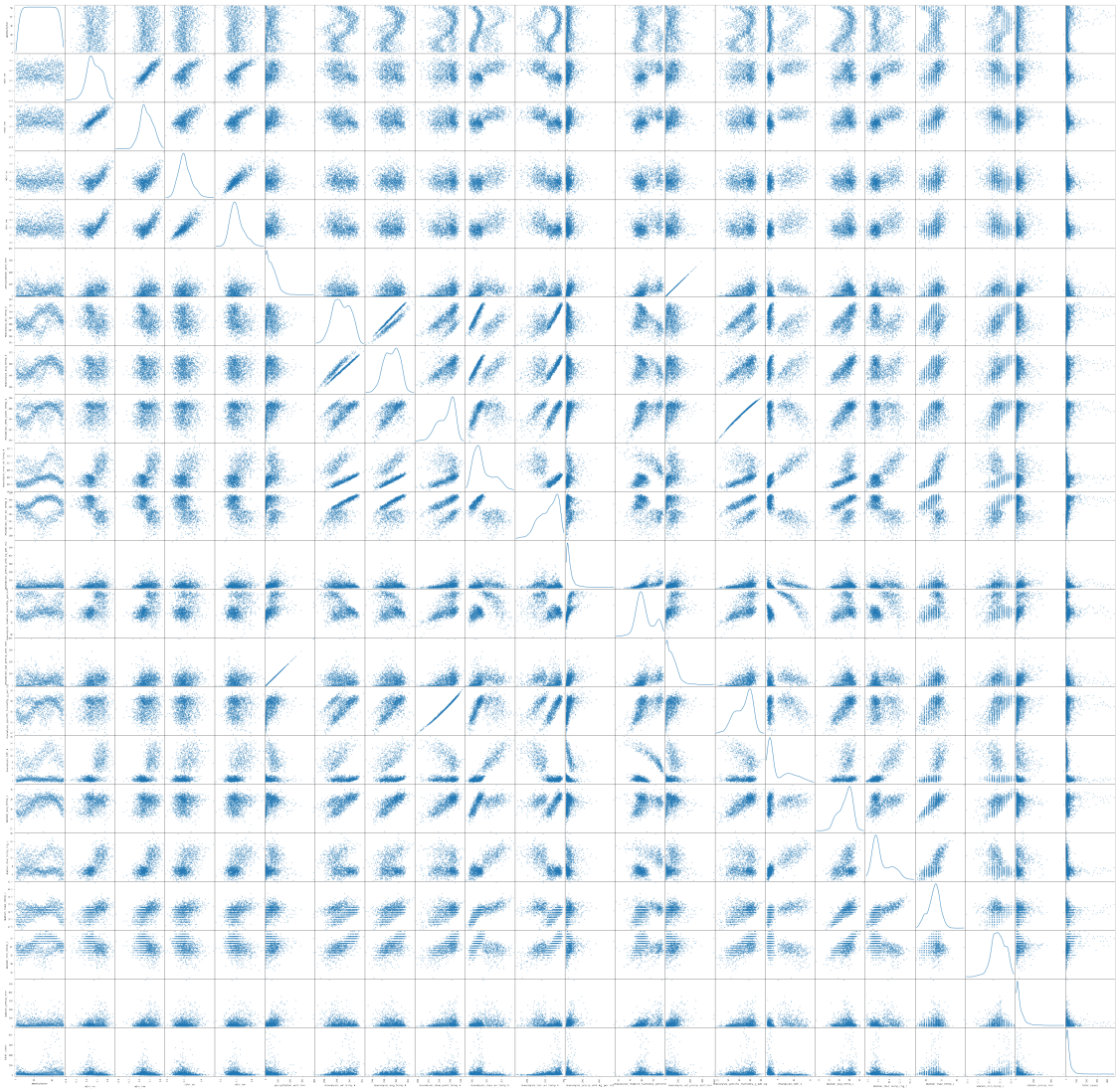


4 Correlations

- There are features which are highly correlated, therefore PCA could become very useful.

- Given that correlations with `total_count` are not very strong, our hunch is that Regression Decision Trees will be more adequate than Linear Regression Models for this problem, because only with the combination of values of different attributes we will obtain an accurate prediction.

[14]:
```
attr = ['weekofyear', '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']
pd.plotting.scatter_matrix(train_data[attr],figsize=(70,70), diagonal='kde',␣
 ↪alpha=0.5)
plt.show()
```



[15]:
```
corr_matrix = train_data[attr].corr()
corr_matrix['total_cases'].sort_values(ascending=False)
```

[15]:
```
total_cases                           1.000000
reanalysis_min_air_temp_k             0.325252
station_min_temp_c                    0.267109
reanalysis_air_temp_k                 0.264952
weekofyear                            0.213117
reanalysis_avg_temp_k                 0.151637
reanalysis_dew_point_temp_k           0.142531
reanalysis_specific_humidity_g_per_kg 0.129861
station_avg_temp_c                    0.116109
reanalysis_precip_amt_kg_per_m2      -0.010031
```

```
reanalysis_sat_precip_amt_mm            -0.038740
precipitation_amt_mm                    -0.038740
station_max_temp_c                      -0.039219
station_precip_mm                       -0.074374
reanalysis_relative_humidity_percent    -0.132452
ndvi_se                                 -0.168612
reanalysis_max_air_temp_k               -0.191345
ndvi_sw                                 -0.196461
ndvi_nw                                 -0.202235
station_diur_temp_rng_c                 -0.237844
ndvi_ne                                 -0.241376
reanalysis_tdtr_k                       -0.278483
Name: total_cases, dtype: float64
```

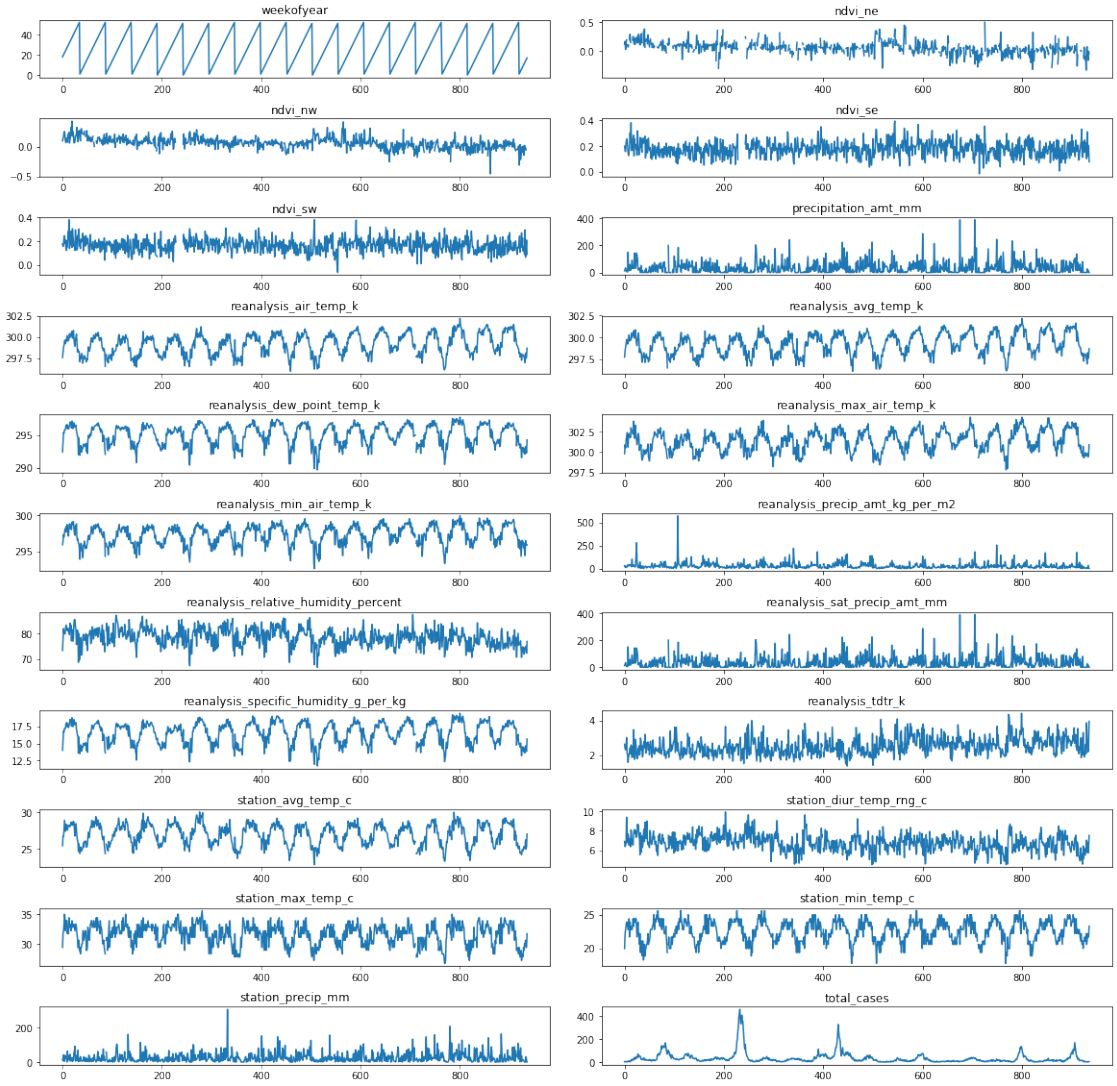
5 Imputation of *NaN* values

- Given that a lot these attributes appear to be relatively continuous, i.e.: it is likely that the weather in a given area does not change a lot over week, using the previous known value **in that city** to imput empty cells might be a better solution than just imputing with the mean or other constant.
- We can confirm this by looking at the line plots of just one city. In some ranges the imputation is clear but we just have to be carefull by not overfitting the model.
- In the case of a first value with *NaN* (i.e.: when there isn't a last value to use as replacement), the `ContinuityImputer` will impute with the median of such attribute.

[16]:
```
train_data[attr].shape
```

[16]:
```
(1451, 22)
```

[17]:
```
f, ax_l = plt.subplots(11, 2, figsize=(20,20))
plt.subplots_adjust(wspace=0.1, hspace=0.7)
data_sj = train_data[train_data['city'] == 'sj']
for ax, at in zip(ax_l.reshape(22), attr):
    ax.plot(data_sj[at])
    ax.set_title(at)
plt.show()
```

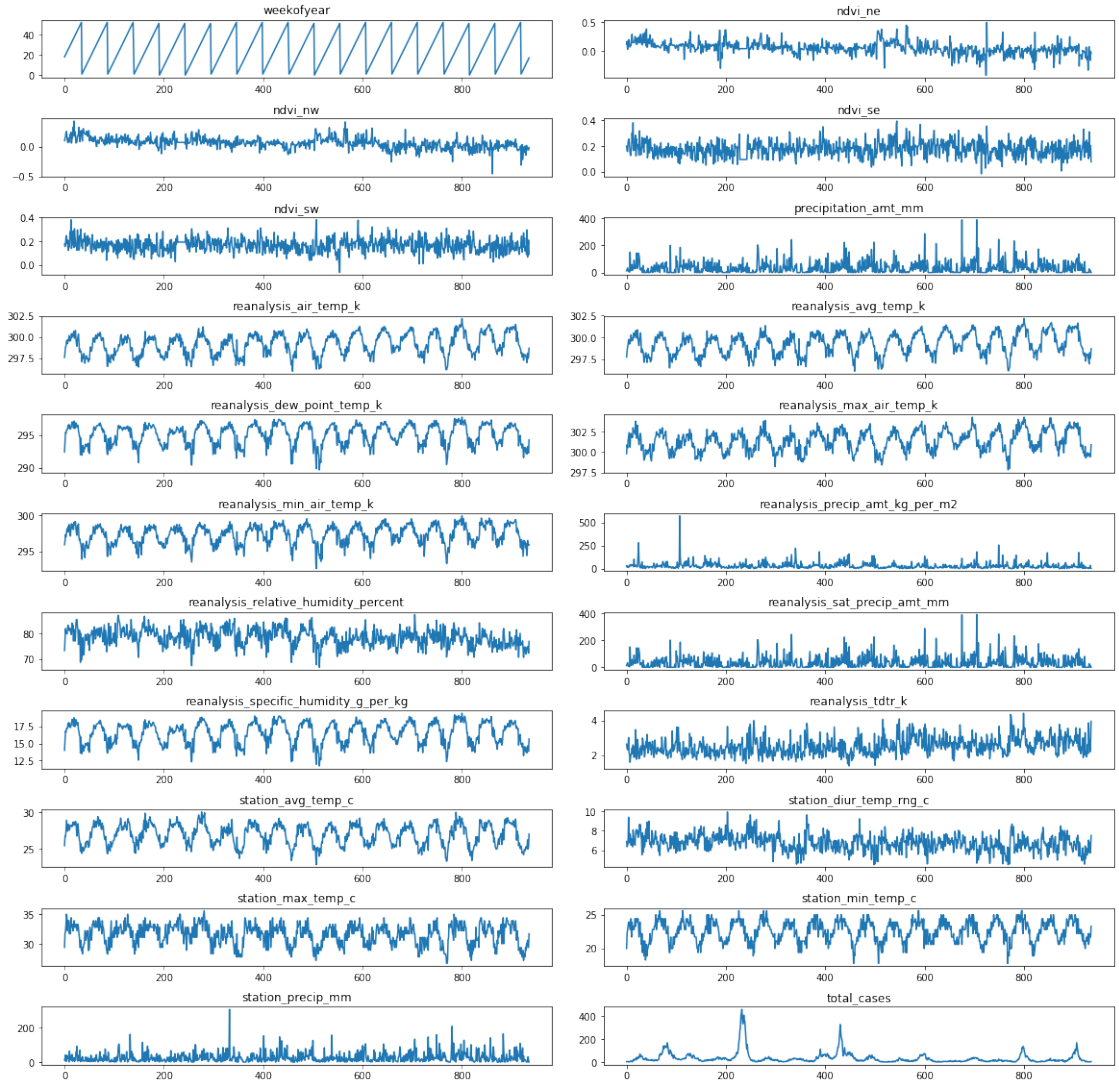


[18]:
```
%autoreload
from utils.ContinuityImputer import ContinuityImputer

imputer = ContinuityImputer(attributes=attr, copy=False)
imputer.fit_transform(train_data)
train_data.isnull().mean() * 100
```

[18]:
```
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
```

[19]:
```
f, ax_l = plt.subplots(11, 2, figsize=(20,20))
plt.subplots_adjust(wspace=0.1, hspace=0.7)
data_sj = train_data[train_data['city'] == 'sj']
for ax, at in zip(ax_l.reshape(22), attr):
    ax.plot(data_sj[at])
    ax.set_title(at)
plt.show()
```



6 Clustering

- To further study the natural grouping of the dataset we will apply clustering techniques and analyze the results.

6.1 Normalization

- In order to apply any distance-based model, previous normalization of the data is necessary. StandardScaler is the best option since it doesn't let outliers deform the data.

[20]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
attr_entity = ['city', 'year', 'weekofyear']
```

```
attr_train = ['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']

SC = ColumnTransformer(transformers=[
    ('norm', StandardScaler(),attr_train),
])
X_train = SC.fit_transform(train_data)
pd.DataFrame(X_train, columns=attr_train).describe()
```

[20]:

	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw
count	1.451000e+03	1.451000e+03	1.451000e+03	1.451000e+03
mean	1.567014e-16	1.958767e-17	-7.835068e-16	5.092794e-16
std	1.000345e+00	1.000345e+00	1.000345e+00	1.000345e+00
min	-3.886308e+00	-4.899542e+00	-2.937297e+00	-3.180207e+00
25%	-6.628943e-01	-6.695191e-01	-6.706252e-01	-6.880688e-01
50%	-1.256285e-01	-1.025250e-01	-9.307602e-02	-1.425495e-01
75%	7.280862e-01	7.118582e-01	6.056024e-01	5.340443e-01
max	2.725022e+00	2.735619e+00	4.521682e+00	4.120158e+00

	precipitation_amt_mm	reanalysis_air_temp_k	reanalysis_avg_temp_k
count	1451.000000	1.451000e+03	1.451000e+03
mean	0.000000	2.460211e-14	-1.499436e-14
std	1.000345	1.000345e+00	1.000345e+00
min	-1.044226	-2.982764e+00	-3.428492e+00
25%	-0.821970	-7.652785e-01	-7.649120e-01
50%	-0.170087	-3.764977e-02	4.943101e-02
75%	0.561492	8.294236e-01	7.789466e-01
max	7.900984	2.568813e+00	2.933563e+00

	reanalysis_dew_point_temp_k	reanalysis_max_air_temp_k
count	1.451000e+03	1.451000e+03
mean	3.614905e-14	1.488663e-15
std	1.000345e+00	1.000345e+00
min	-3.668365e+00	-1.739303e+00
25%	-7.369308e-01	-7.496538e-01
50%	2.579885e-01	-3.166825e-01
75%	7.949737e-01	6.420397e-01
max	2.099080e+00	3.270794e+00

	reanalysis_min_air_temp_k	reanalysis_precip_amt_kg_per_m2
count	1.451000e+03	1.451000e+03

```
mean                 5.876301e-15                    -9.793835e-17
std                  1.000345e+00                     1.000345e+00
min                 -3.442197e+00                    -9.251965e-01
25%                 -7.097565e-01                    -6.238675e-01
50%                  1.880453e-01                    -2.981001e-01
75%                  8.516378e-01                     2.782753e-01
max                  1.632335e+00                     1.222769e+01

       reanalysis_relative_humidity_percent  reanalysis_sat_precip_amt_mm  \
count                          1.451000e+03                   1451.000000
mean                          -1.175260e-15                      0.000000
std                            1.000345e+00                      1.000345
min                           -3.408184e+00                     -1.044226
25%                           -6.965726e-01                     -0.821970
50%                           -2.605496e-01                     -0.170087
75%                            5.797384e-01                      0.561492
max                            2.299462e+00                      7.900984

       reanalysis_specific_humidity_g_per_kg  reanalysis_tdtr_k  \
count                           1.451000e+03       1.451000e+03
mean                           -2.497428e-15       7.835068e-17
std                             1.000345e+00       1.000345e+00
min                            -3.262159e+00      -1.000323e+00
25%                            -7.693771e-01      -7.261981e-01
50%                             2.212490e-01      -5.770417e-01
75%                             7.990371e-01       7.673817e-01
max                             2.411005e+00       3.139775e+00

       station_avg_temp_c  station_diur_temp_rng_c  station_max_temp_c  \
count        1.451000e+03             1.451000e+03        1.451000e+03
mean        -2.742274e-16             1.567014e-16       -1.136085e-15
std          1.000345e+00             1.000345e+00        1.000345e+00
min         -4.512338e+00            -1.676704e+00       -2.933447e+00
25%         -6.884901e-01            -7.349616e-01       -6.880898e-01
50%          1.699247e-01            -3.380845e-01        1.794344e-01
75%          7.496333e-01             7.112856e-01        7.407736e-01
max          2.823207e+00             3.630687e+00        4.976333e+00

       station_min_temp_c  station_precip_mm
count        1.451000e+03       1.451000e+03
mean        -1.958767e-15       9.793835e-17
std          1.000345e+00       1.000345e+00
min         -4.709859e+00      -8.266708e-01
25%         -6.361639e-01      -6.430863e-01
50%          6.400242e-02      -3.265611e-01
75%          7.641687e-01       3.043789e-01
max          2.228153e+00       1.063787e+01
```

6.2 K-means

6.2.1 Elbow Method

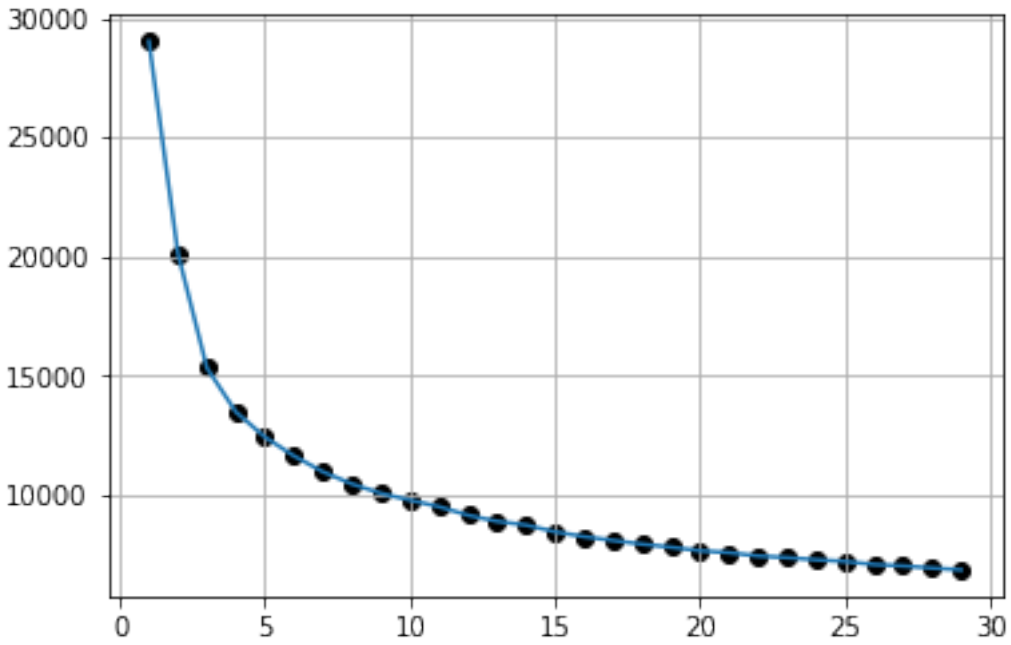
- In order to find the "optimal" number of clusters in K-means we will apply the elbow method.
- The result of the elbow method was that the optimal number of clusters should be 2.

[21]:
```
from sklearn.cluster import KMeans
all_kmeans = [KMeans(n_clusters=n, init='k-means++', n_init=10, max_iter=300,␣
↪random_state=42).fit(X_train) for n in range(1, 30)]
```

[22]:
```
#Elbow method
best_n = 2
best_delta = -np.inf
for idx, km in enumerate(all_kmeans):
    if idx + 1 >= 2:
        delta = previous_value - km.inertia_
        if delta > best_delta:
            best_delta = delta
            best_n = idx + 1
            best_km = km
    previous_value = km.inertia_

best_n
```

[22]: 2

[23]:
```
inertias = list(map(lambda x: x.inertia_, all_kmeans))
plt.grid(True)
plt.scatter(x=range(1,30), y=inertias, color='k')
plt.plot(range(1,30), inertias)
plt.show()
```

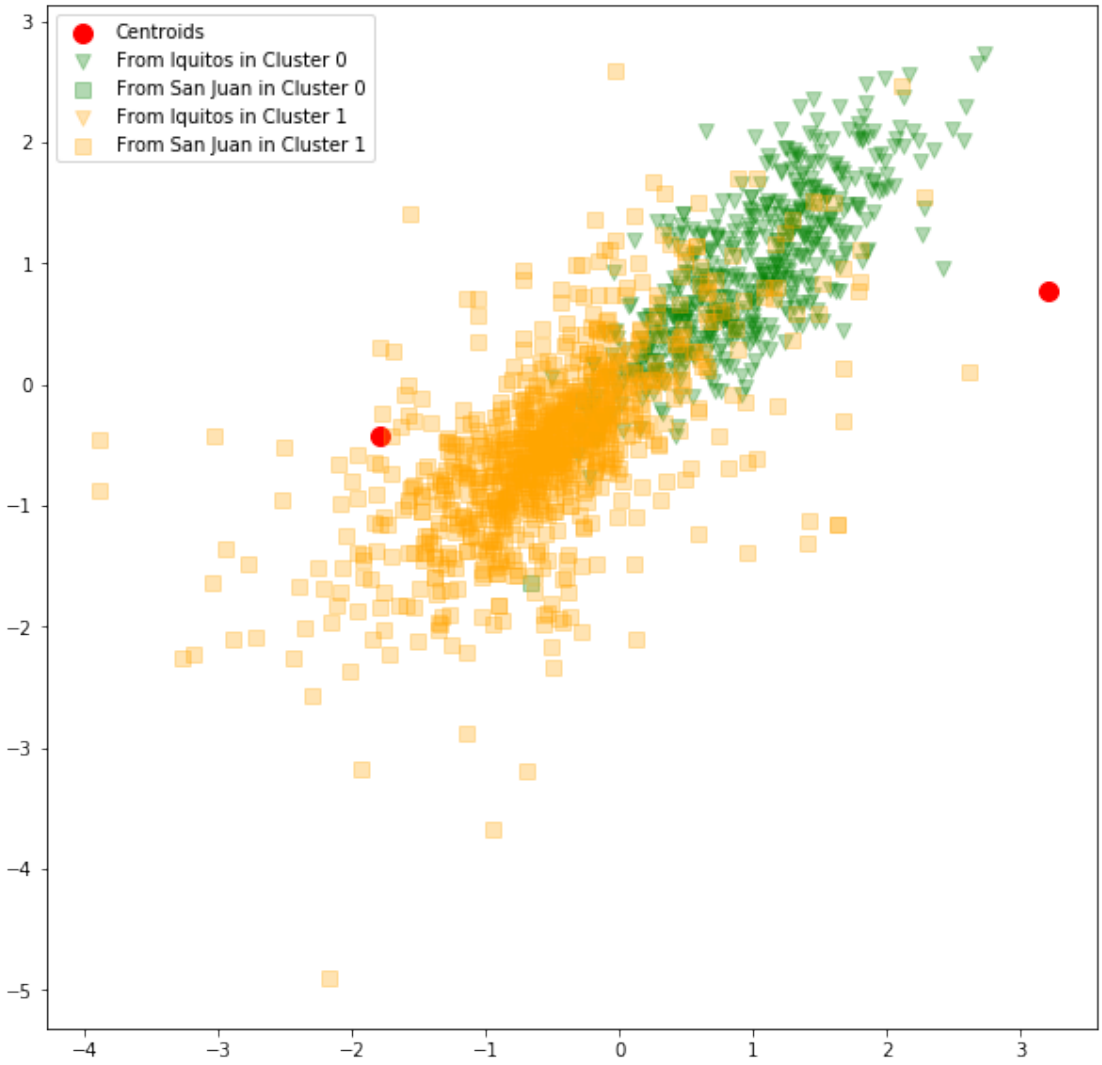


6.2.2 Visualisation

- We will apply PCA to the data in order to visualise the clusters.
- It is very interesting to note that K-means is detecting with very high accuracy from which city is the data, as we can see from the scatter plot below. The fact that the elbow method decided that 2 clusters was the ideal number of groups to split the data and that K-means is splitting it into the 2 existing cities is a great indicator that the weather and the NDVI readings from both places are very distinct.

[24]:
```
from sklearn.decomposition import PCA

labels = best_km.predict(X_train)

pca = PCA(n_components=2)
train_pca = pca.fit_transform(X_train)
centroid_pca = pca.transform(best_km.cluster_centers_)

X_iq = X_train[train_data['city'] == 'iq']
labels_iq = labels[train_data['city'] == 'iq']
X_sj = X_train[train_data['city'] == 'sj']
labels_sj = labels[train_data['city'] == 'sj']

def clusterIndexFromLabels(labels, cluster): return list(map(lambda x:␣
 ↪x==cluster, labels))
```

```
clusters = {0: 'green', 1: 'orange'}

plt.figure(figsize=(10,10))

plt.scatter(centroid_pca[:,0], centroid_pca[:,1], s=100, color='red')
legends=['Centroids']

for i in range(best_n):
    index = clusterIndexFromLabels(labels_iq, i)
    plt.scatter(X_iq[index, 0], X_iq[index,1], s=60, marker='v',␣
 ↪color=clusters[i], alpha=0.3)
    legends.append('From Iquitos in Cluster ' + str(i))

    index = clusterIndexFromLabels(labels_sj, i)
    plt.scatter(X_sj[index, 0], X_sj[index,1], s=60, marker='s',␣
 ↪color=clusters[i], alpha=0.3)
    legends.append('From San Juan in Cluster ' + str(i))

plt.legend(legends)

plt.show()

print('Fraction of Iquitos instances in cluster 1: ', labels_iq.mean())
print('Fraction of San Juan instances in cluster 1: ', labels_sj.mean())
```




```
Fraction of Iquitos instances in cluster 1:  0.0
Fraction of San Juan instances in cluster 1:  0.9978563772775991
```

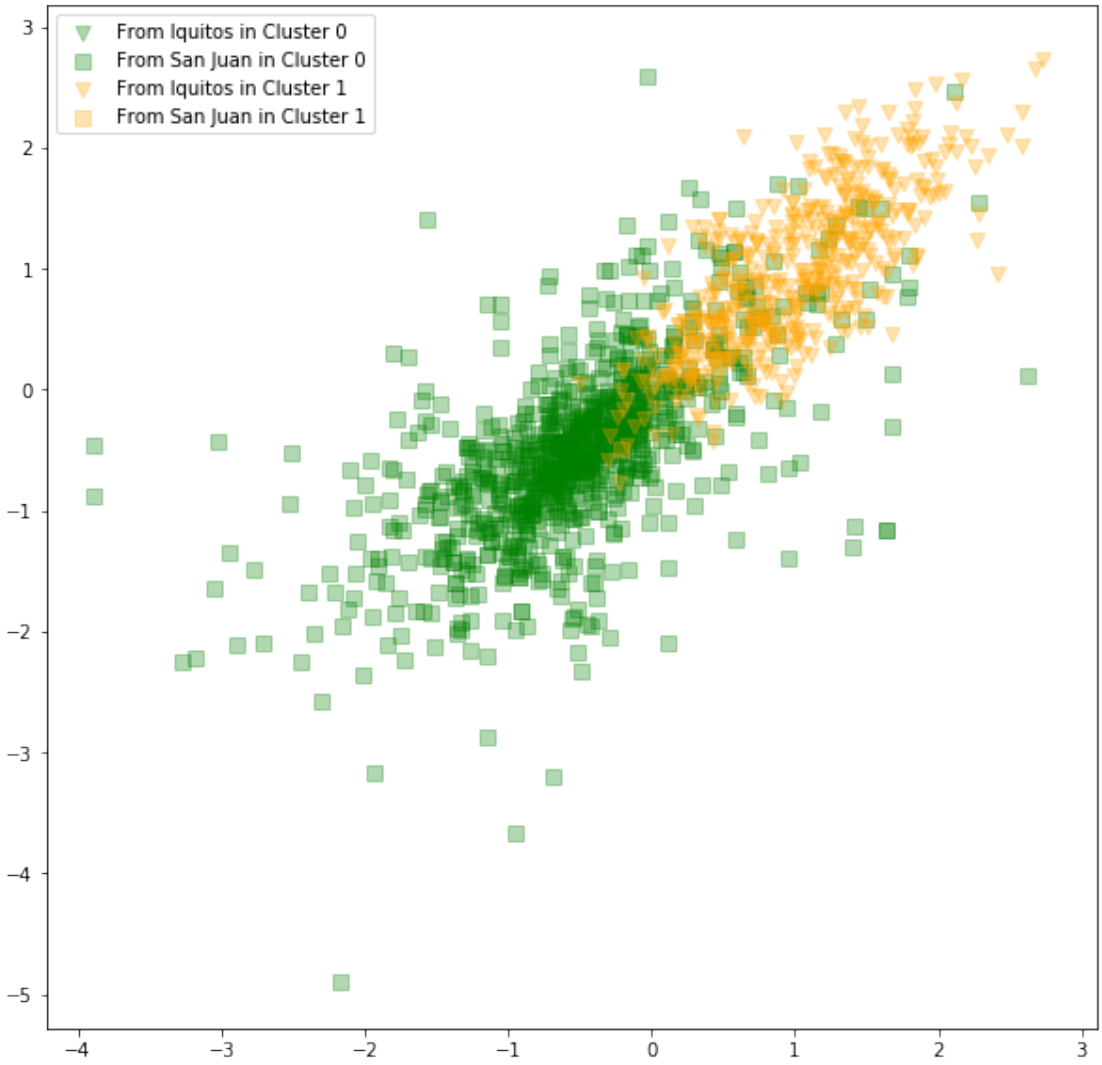
6.3 Hierarchical Clustering

- Hierarchical clustering is pratically as good distinguishing the cities.

[25]:
```
from sklearn.cluster import AgglomerativeClustering

hier = AgglomerativeClustering(n_clusters=2, linkage='ward')
hier.fit(X_train)
```

[25]:
```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
            connectivity=None, linkage='ward', memory=None, n_clusters=2,
            pooling_func='deprecated')
```

[26]:
```
labels = hier.labels_

pca = PCA(n_components=2)
train_pca = pca.fit_transform(X_train)

X_iq = X_train[train_data['city'] == 'iq']
labels_iq = labels[train_data['city'] == 'iq']
X_sj = X_train[train_data['city'] == 'sj']
labels_sj = labels[train_data['city'] == 'sj']

def clusterIndexFromLabels(labels, cluster): return list(map(lambda x:␣
↪x==cluster, labels))

clusters = {0: 'green', 1: 'orange'}

plt.figure(figsize=(10,10))

legends=[]

for i in range(2):
    index = clusterIndexFromLabels(labels_iq, i)
    plt.scatter(X_iq[index, 0], X_iq[index,1], s=60, marker='v',␣
↪color=clusters[i], alpha=0.3)
    legends.append('From Iquitos in Cluster ' + str(i))

    index = clusterIndexFromLabels(labels_sj, i)
    plt.scatter(X_sj[index, 0], X_sj[index,1], s=60, marker='s',␣
↪color=clusters[i], alpha=0.3)
    legends.append('From San Juan in Cluster ' + str(i))

plt.legend(legends)

plt.show()

print('Fraction of Iquitos instances in cluster 1: ', labels_iq.mean())
print('Fraction of San Juan instances in cluster 1: ', labels_sj.mean())
```




```
Fraction of Iquitos instances in cluster 1:  0.9980694980694981
Fraction of San Juan instances in cluster 1:  0.0010718113612004287
```

7 Attribute by attribute analysis

7.0.1 Time & City

- The time interval is very big. Older instances might be a problem for predictor models to train on.
- Week of the year appears to have correlation, however that should be because it is already correlated with the weather, so adding it to the predictive model probably wouldn't increase its performance.

[27]:
```
min(train_data[train_data['city'] == 'iq']['week_start_date']),␣
↪max(train_data[train_data['city'] == 'iq']['week_start_date'])
```

[27]:
```
('2000-07-01', '2010-06-25')
```

[28]:
```
min(test_data[test_data['city'] == 'iq']['week_start_date']),␣
↪max(test_data[test_data['city'] == 'iq']['week_start_date'])
```

[28]:
```
('2010-07-02', '2013-06-25')
```

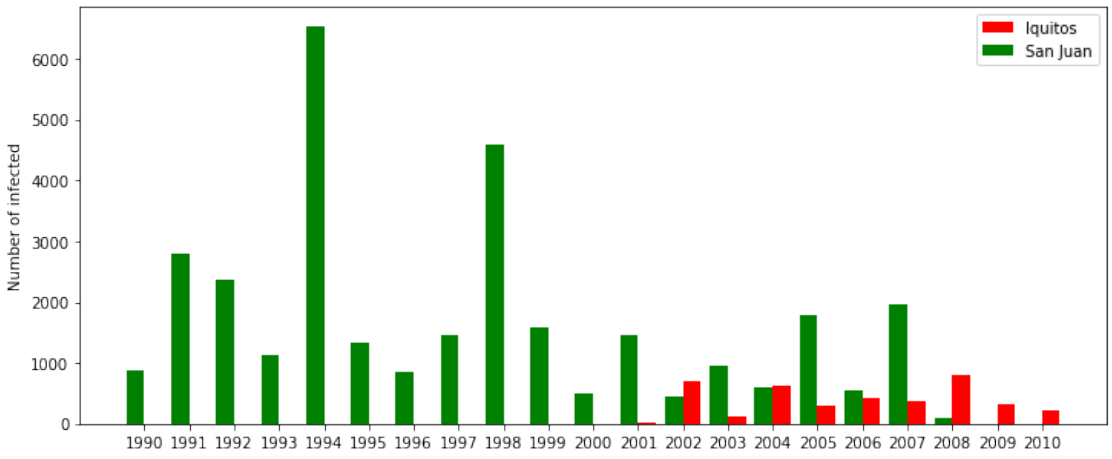
[29]:
```
min(train_data[train_data['city'] == 'sj']['week_start_date']),␣
↪max(train_data[train_data['city'] == 'sj']['week_start_date'])
```

[29]:
```
('1990-04-30', '2008-04-22')
```

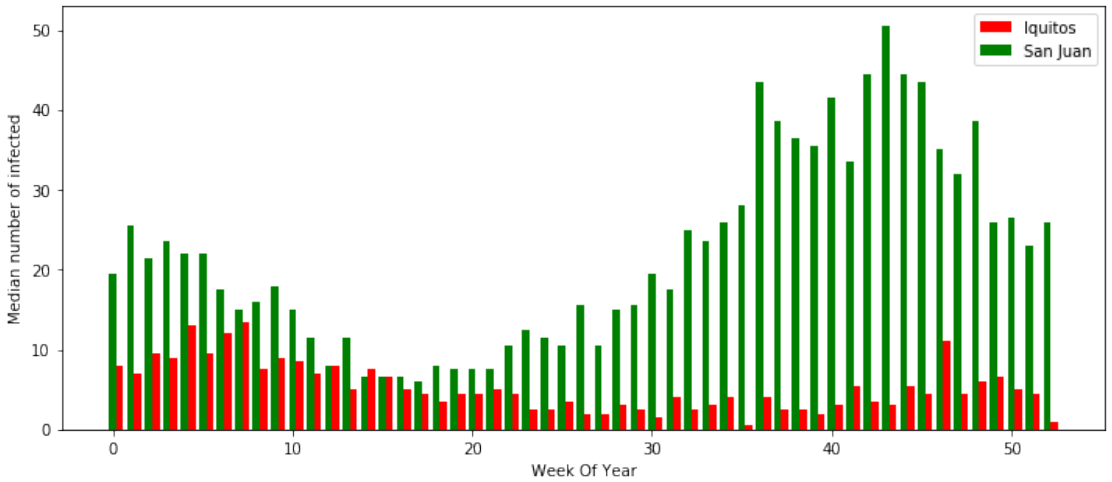
[30]:
```
min(test_data[test_data['city'] == 'sj']['week_start_date']),␣
↪max(test_data[test_data['city'] == 'sj']['week_start_date'])
```

[30]:
```
('2008-04-29', '2013-04-23')
```

[31]:
```
cases_by_year_sj = train_data[train_data['city'] ==␣
↪'sj'][['year','total_cases']].groupby(by=['year']).sum()
cases_by_year_iq = train_data[train_data['city'] ==␣
↪'iq'][['year','total_cases']].groupby(by=['year']).sum()
plt.figure(figsize=(12,5))
w=0.4
plt.bar(cases_by_year_iq.index.values+w, width=w,␣
↪height=cases_by_year_iq['total_cases'], align='center', color='red')
plt.bar(cases_by_year_sj.index.values, width=w,␣
↪height=cases_by_year_sj['total_cases'], align='center', color='green')
plt.legend(labels=['Iquitos', 'San Juan'])
plt.ylabel('Number of infected')
years = np.array(range(1990, 2011))
plt.xticks(years+w/2, years)
plt.show()
```



[32]:
```
cases_by_week_sj = train_data[train_data['city'] ==␣
 ↪'sj'][['weekofyear','total_cases']].groupby(by=['weekofyear']).median()
cases_by_week_iq = train_data[train_data['city'] ==␣
 ↪'iq'][['weekofyear','total_cases']].groupby(by=['weekofyear']).median()
plt.figure(figsize=(12,5))
w=0.4
plt.bar(cases_by_week_iq.index.values+w, width=w,␣
 ↪height=cases_by_week_iq['total_cases'], align='center', color='red')
plt.bar(cases_by_week_sj.index.values, width=w,␣
 ↪height=cases_by_week_sj['total_cases'], align='center', color='green')
plt.legend(labels=['Iquitos', 'San Juan'])
plt.ylabel('Median number of infected')
plt.xlabel('Week Of Year')
plt.show()
```



7.0.2 NDVI

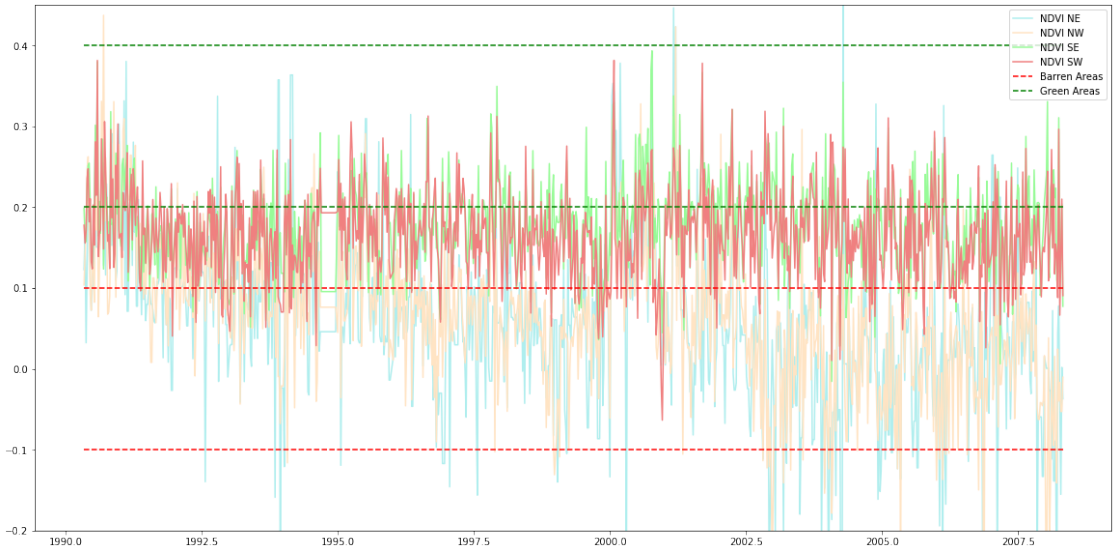
- *"Negative values of NDVI (values approaching -1) correspond to water. Values close to zero (-0.1 to 0.1) generally correspond to barren areas of rock, sand, or snow. Lastly, low, positive values represent shrub and grassland (approximately 0.2 to 0.4), while high values indicate temperate and tropical rainforests (values approaching 1)"*
- It appears to make sense to imput missing NDVI values with the previous ones, because it shouldn't change much over week.

San Juan

[33]:
```
ndvi_sj = train_data[train_data['city'] == 'sj'][['year',␣
↪'weekofyear','ndvi_ne', 'ndvi_nw', 'ndvi_se', 'ndvi_sw']]
```

The space between the 2 green dashed lines represents that the NDVI measure was one of a green space. Between the 2 red dashed lines the reading represented barren areas.

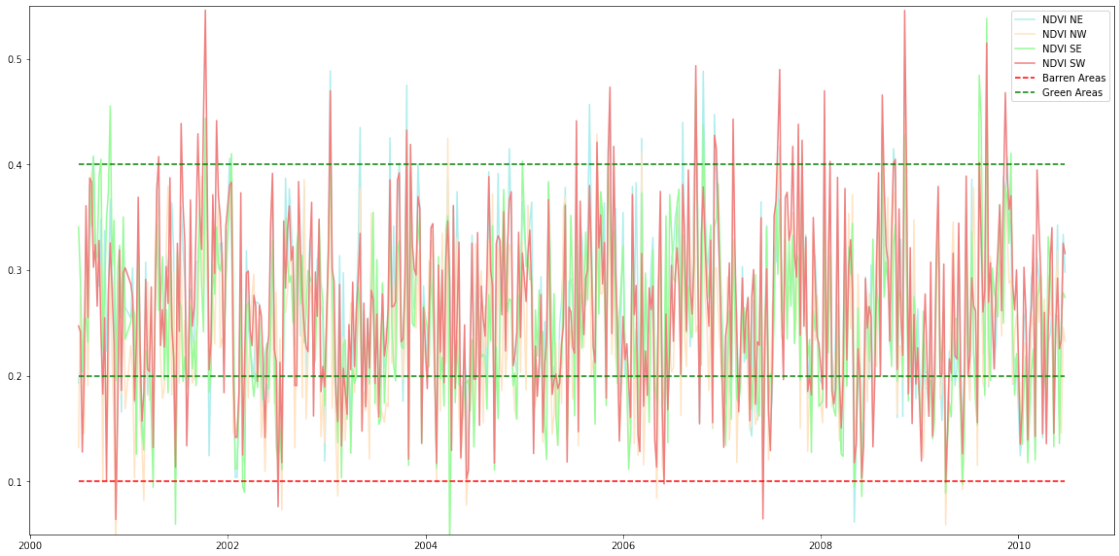
[34]:
```
# 52 is the number of weeks per year
x = ndvi_sj['year'] + (ndvi_sj['weekofyear'] / 53.0)
ne = ndvi_sj['ndvi_ne']
nw = ndvi_sj['ndvi_nw']
se = ndvi_sj['ndvi_se']
sw = ndvi_sj['ndvi_sw']
plt.figure(figsize=(20,10))
plt.ylim(-0.2, 0.45)
plt.plot(x, ne, color='paleturquoise')
plt.plot(x, nw, color='bisque')
plt.plot(x, se, color='palegreen')
plt.plot(x, sw, color='lightcoral')
plt.hlines(-0.1, xmin=x[0], xmax=x.iloc[-1], linestyles='dashed', colors='red',␣
↪zorder=4)
plt.hlines(0.2, xmin=x[0], xmax=x.iloc[-1], linestyles='dashed',␣
↪colors='green', zorder=4)
plt.hlines(0.1, xmin=x[0], xmax=x.iloc[-1], linestyles='dashed', colors='red',␣
↪zorder=4)
plt.hlines(0.4, xmin=x[0], xmax=x.iloc[-1], linestyles='dashed',␣
↪colors='green', zorder=4)
labels=['NDVI NE', 'NDVI NW', 'NDVI SE', 'NDVI SW', 'Barren Areas', 'Green␣
↪Areas']
plt.legend(labels=labels)
plt.show()
```



Iquitos

[35]:
```
ndvi_iq = train_data[train_data['city'] == 'iq'][['year',␣
 ↪'weekofyear','ndvi_ne', 'ndvi_nw', 'ndvi_se', 'ndvi_sw']]
```

[36]:
```
# 52 is the number of weeks per year
x = ndvi_iq['year'] + (ndvi_iq['weekofyear'] / 53.0)
ne = ndvi_iq['ndvi_ne']
nw = ndvi_iq['ndvi_nw']
se = ndvi_iq['ndvi_se']
sw = ndvi_iq['ndvi_sw']
plt.figure(figsize=(20,10))
plt.ylim(0.05, 0.55)
plt.plot(x, ne, color='paleturquoise')
plt.plot(x, nw, color='bisque')
plt.plot(x, se, color='palegreen')
plt.plot(x, sw, color='lightcoral')
plt.hlines(-0.1, xmin=x.iloc[0], xmax=x.iloc[-1], linestyles='dashed',␣
 ↪colors='red', zorder=4)
plt.hlines(0.2, xmin=x.iloc[0], xmax=x.iloc[-1], linestyles='dashed',␣
 ↪colors='green', zorder=4)
plt.hlines(0.1, xmin=x.iloc[0], xmax=x.iloc[-1], linestyles='dashed',␣
 ↪colors='red', zorder=4)
plt.hlines(0.4, xmin=x.iloc[0], xmax=x.iloc[-1], linestyles='dashed',␣
 ↪colors='green', zorder=4)
labels=['NDVI NE', 'NDVI NW', 'NDVI SE', 'NDVI SW', 'Barren Areas', 'Green␣
 ↪Areas']
plt.legend(labels=labels)
plt.show()
```



8 PCA

- PCA must clearly be applied before predicting, given the strong correlation between attributes.

[57]:
```
%autoreload
from OurPipeline import create_pipeline
from sklearn.decomposition import PCA

attr=list(train_data)[:-1]
pipeline = create_pipeline(attr, n_weeks=3, pca=PCA(0.999))

X_train = pipeline.fit_transform(train_data.iloc[:,:-1].copy(), train_data.
 ↪iloc[:,-1].copy())
X_train.shape
```

[57]:
```
(1451, 65)
```

[63]:
```
pca = pipeline.named_steps['pca']
m = len(pca.explained_variance_ratio_)
plt.figure(figsize=(20,5))
plt.bar(x=range(m), height=pca.explained_variance_ratio_ * 100)
plt.xticks(range(m), [str(i) for i in range(1,m+1)])
plt.title("Explained variance")
plt.ylabel("Percentage")

explained_total = np.ndarray(shape=[m])
curr = 0
```

```
for idx, var in enumerate(pca.explained_variance_ratio_):
    curr += var
    explained_total[idx] = curr
    plt.annotate(str(int(curr*100)), (idx-0.5, curr*100 + 1.2))

plt.plot(range(m), explained_total * 100, color='red')
plt.legend(['Sum of explained variances'])
plt.scatter(x=range(m), y=explained_total * 100, zorder=5, color='k', s=10)
plt.grid()
plt.show()
```