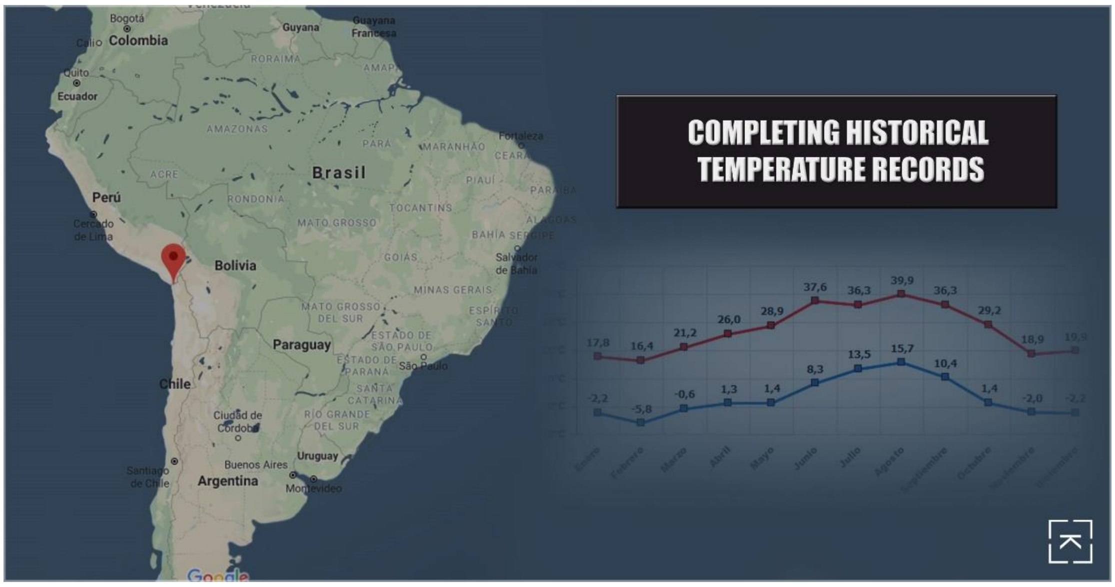


Completing historical temperature records

BASED ON PUBLIC DATA FROM L.A.C.A. WE COMPLETED THE HISTORICAL TEMPERATURE RECORD OF THE AZAPA-CHILE WEATHER STATION BETWEEN 1977-1980.

Gabriel Naya | Kreilabs | 10/6/2019

My intention in this exercise was to locate a set of data on the climate of Uruguay, to review the evolution of the daily average temperature marks registered in different points of the country, and -applying artificial intelligence- complete the records in case there were missing data.

It was impossible for me to find a public dataset of Uruguay, but I could access data published by Latin American Climate Assessment & Dataset (LACA&D), obtaining TXT files by consulting the site: http://lacad.ciifen.org/ES/.

Problem Formulation

The exercise consists of delimiting the data set to the weather stations of a region (Chile), looking for files that report records in similar periods of time (in years).

Once the data from at least 6 weather stations have been obtained, we select one as a destination, and try to find a way to predict the values of the missing records.

The dataset

Records were pre-selected and obtained from the following stations:

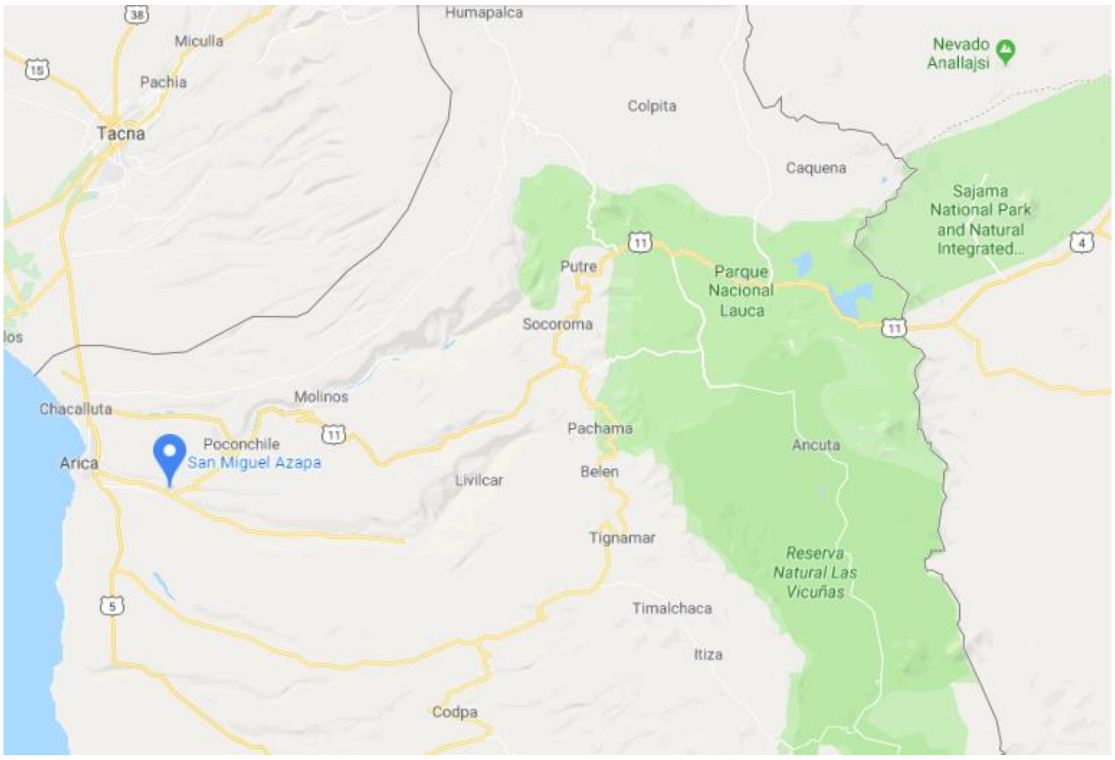
STAID	STANAME	CN	LAT	LON	HGHT
553	CAQUENA	CL	-18:03:15	-69:12:06	4400
557	CHUNGARA AJATA	CL	-18:14:07	-69:11:00	4585
562	PARINACOTA EX ENDESA	CL	-18:12:15	-69:16:06	4420
564	GUALLATIRE	CL	-18:29:54	-69:09:17	4240
565	CHILCAYA	CL	-18:47:38	-69:05:04	4270
584	PACOLLO	CL	-18:10:37	-69:30:33	4185
585	PUTRE	CL	-18:11:57	-69:33:37	3545
587	PUTRE (DCP)	CL	-18:11:42	-69:33:32	3560
588	LLUTA	CL	-18:24:37	-70:10:09	290
589	MURMUNTANE	CL	-18:21:07	-69:33:07	3550
595	ARICA OFICINA	CL	-18:28:39	-70:19:15	20
596	AZAPA	CL	-18:30:56	-70:10:50	365
597	U. DEL NORTE	CL	-18:29:00	-70:17:37	55
598	EL BUITRE AERODROMO	CL	-18:30:43	-70:17:03	110
599	CHACA	CL	-18:49:01	-70:09:00	350
600	CODPA	CL	-18:49:56	-69:44:38	1870

Among them, after analyzing the reported periods and the quality of the imputed data, the following stations in northern Chile and southern Peru were definitively selected as input information support:

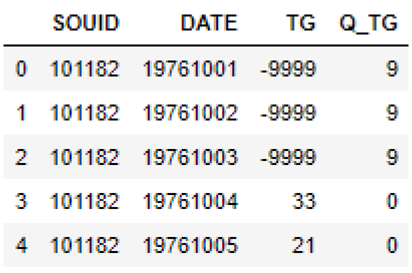
STAID	STANAME	CN	LAT	LON	HGHT
553	CAQUENA	CL	-18:03:15	-69:12:06	4400
585	PUTRE	CL	-18:11:57	-69:33:37	3545
588	LLUTA	CL	-18:24:37	-70:10:09	290
595	ARICA OFICINA	CL	-18:28:39	-70:19:15	20
597	U. DEL NORTE	CL	-18:29:00	-70:17:37	55

And the AZAPA weather station (596) was selected as the target.

STAID	STANAME	CN	LAT	LON	HGHT
596	AZAPA	CL	-18:30:56	-70:10:50	365

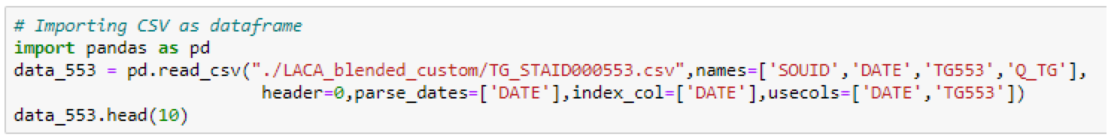


Basically all files have a date (DATE), a source id (SOUID), the temperature in tenths of degrees Celsius (TG) and a quality data indicator (Q_TG), where Q_TG=zero implies a real data, and Q_TG=9 implies a manually completed data.

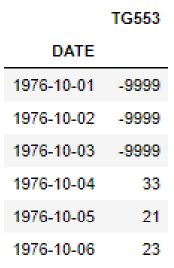


	SOUID	DATE	TG	Q_TG
0	101182	19761001	-9999	9
1	101182	19761002	-9999	9
2	101182	19761003	-9999	9
3	101182	19761004	33	0
4	101182	19761005	21	0

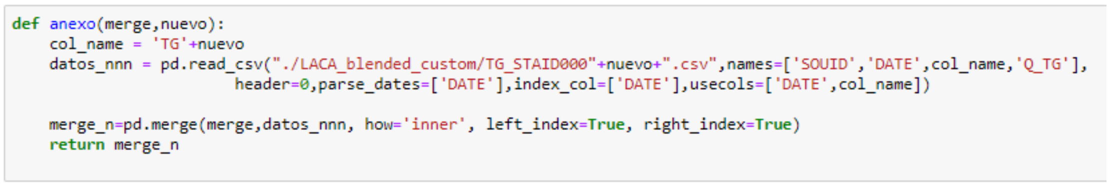
We are going to import the date as index, and only the TG column, creating each of the data frames and joining them by the date index:



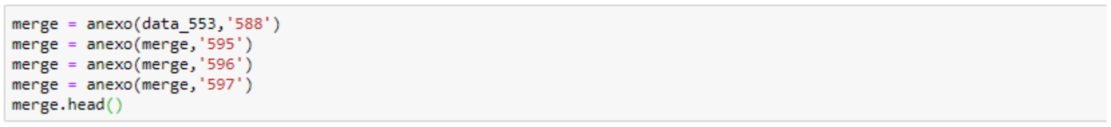
```
# Importing CSV as dataframe
import pandas as pd
data_553 = pd.read_csv("./LACA_blended_custom/TG_STAID000553.csv",names=['SOUID','DATE','TG553','Q_TG'],
                       header=0,parse_dates=['DATE'],index_col=['DATE'],usecols=['DATE','TG553'])
data_553.head(10)
```



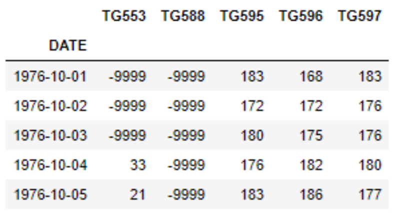
	TG553
DATE	
1976-10-01	-9999
1976-10-02	-9999
1976-10-03	-9999
1976-10-04	33
1976-10-05	21
1976-10-06	23



```
def anexo(merge,nuevo):
    col_name = 'TG'+nuevo
    datos_nnn = pd.read_csv("./LACA_blended_custom/TG_STAID000"+nuevo+".csv",names=['SOUID','DATE',col_name,'Q_TG'],
                       header=0,parse_dates=['DATE'],index_col=['DATE'],usecols=['DATE',col_name])

    merge_n=pd.merge(merge,datos_nnn, how='inner', left_index=True, right_index=True)
    return merge_n
```



```
merge = anexo(data_553,'588')
merge = anexo(merge,'595')
merge = anexo(merge,'596')
merge = anexo(merge,'597')
merge.head()
```



	TG553	TG588	TG595	TG596	TG597
DATE					
1976-10-01	-9999	-9999	183	168	183
1976-10-02	-9999	-9999	172	172	176
1976-10-03	-9999	-9999	180	175	176
1976-10-04	33	-9999	176	182	180
1976-10-05	21	-9999	183	186	177

At this point we have a dataframe with the integrated data set to start the different approaches to a solution, let's go for it!

Data pre-view and feature engineering

The first thing we are going to do is to assign as null the registers that have TG=-9999

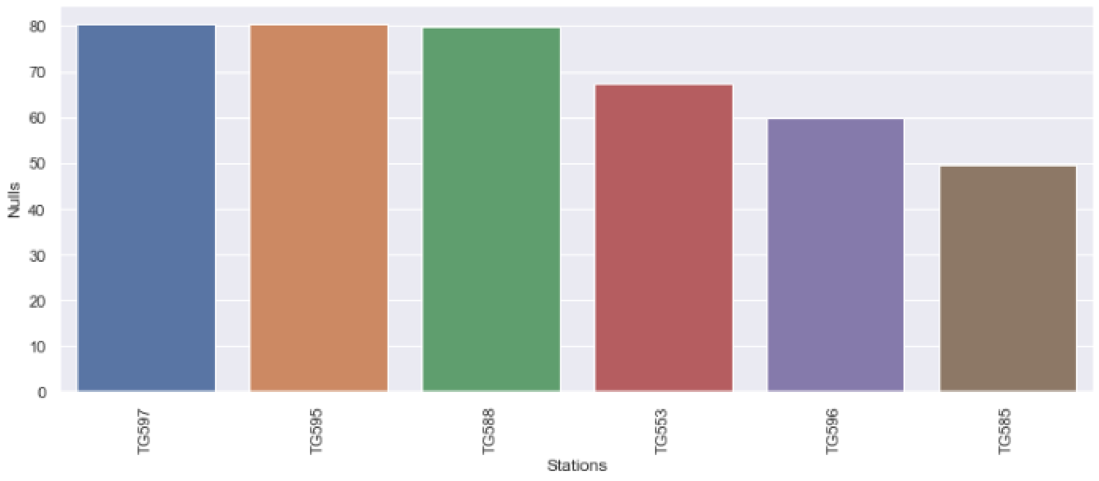
```
#Asignamos None a los registros con -9999
cols = ['TG553','TG588','TG585','TG595','TG596','TG597']
for col in cols:
    merge[col] = merge[col].apply(lambda x: None if x == -9999 else x)

merge.head()
```

	TG553	TG585	TG588	TG595	TG596	TG597
DATE						
1976-10-01	NaN	NaN	NaN	183.0	168.0	183.0
1976-10-02	NaN	NaN	NaN	172.0	172.0	176.0
1976-10-03	NaN	NaN	NaN	180.0	175.0	176.0
1976-10-04	33.0	NaN	NaN	176.0	182.0	180.0
1976-10-05	21.0	NaN	NaN	183.0	186.0	177.0

And display the null data by column

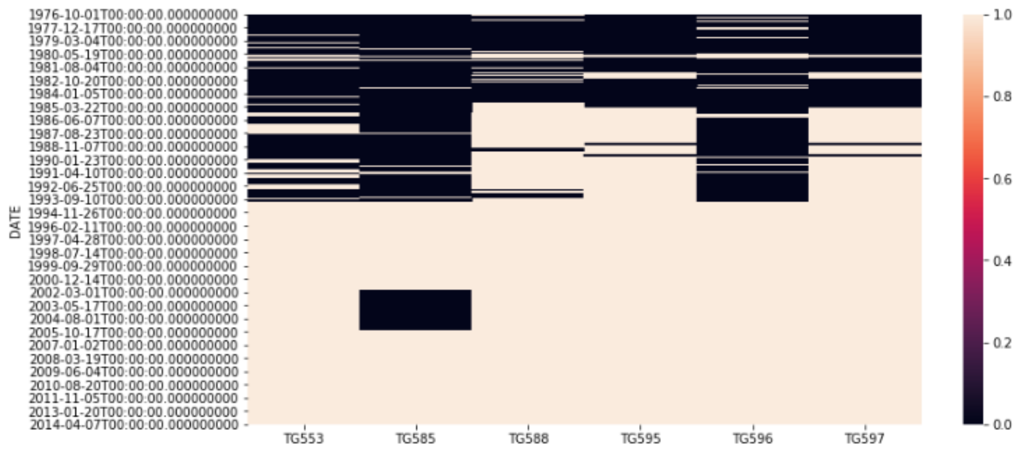
```
merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 14122 entries, 1976-10-01 to 2015-05-31
Data columns (total 6 columns):
TG553     4620 non-null float64
TG585     7128 non-null float64
TG588     2857 non-null float64
TG595     2742 non-null float64
TG596     5664 non-null float64
TG597     2742 non-null float64
dtypes: float64(6)
memory usage: 772.3 KB
```



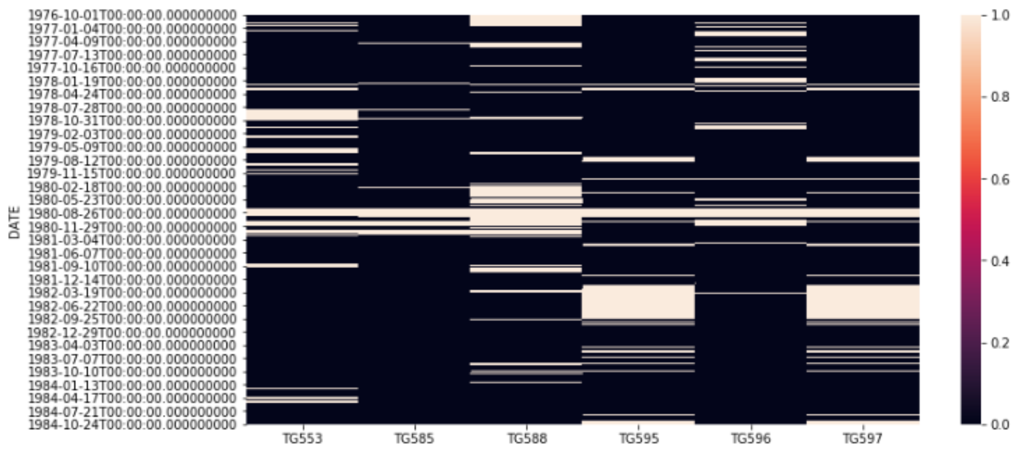
If we analyze the missing information by period in a heatmap:

```
plt.figure(figsize=(12, 6))
sns.heatmap(merge.isnull())
plt.show()
```



We can observe that the information (black background) is quite compact in the period October-1976 to December 1984.

```
plt.figure(figsize=(12, 6))
sns.heatmap(merge.loc['1976-10-01':'1984-12-31'].isnull())
plt.show()
```



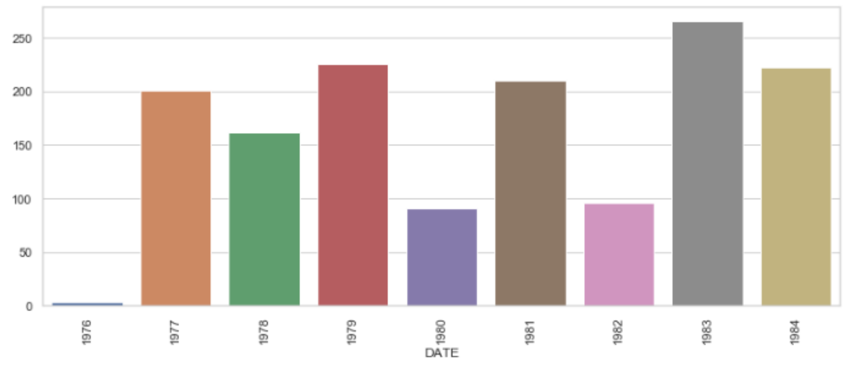
When applying the elimination of null records in the whole dataset we are left with a data set of 1477 records, located between 12/28/1976 and 11/21/1984.

```
merge.dropna(inplace=True)

merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1477 entries, 1976-12-28 to 1984-11-21
Data columns (total 6 columns):
TG553     1477 non-null float64
TG585     1477 non-null float64
TG588     1477 non-null float64
TG595     1477 non-null float64
TG596     1477 non-null float64
TG597     1477 non-null float64
dtypes: float64(6)
memory usage: 80.8 KB
```

Deploying data by year:

```
anual = merge.groupby(merge.index.year)
serie = anual.size()

plt.figure(figsize=(13, 5))
sns.set(style='whitegrid')
sns.barplot(x=serie.index, y=serie)
plt.xticks(rotation = 90)
plt.show()
```



Let's go for the approaches to the solution.

Strategy

Always before approaching a problem, I like design a strategy. In fact, I believe that the strategy is in my head, but before fully developing it, is very useful to make it explicit and then come back to it. As much as we know that as we explore solutions, new elements emerge that make us move away from this original idea, make it vary, it is good to write it before starting, and modify it at the end of the work.

About this exercise, my original idea was very modest, I started it as a Saturday morning exercise to keep me in shape, and it has really challenged me more than I expected:

- Let's see, this is a regression problem, we must obtain the temperature of one of the stations, so let's try regression algorithms and surely the job is finished.
- The data source is homogeneous, we wouldn't need to do many things with features.
- The data set is wide enough to think about good results, it does not present imbalances in the data periods, or missing in certain months, things of the style that lead us to think about data balancing.
- If the regression algorithms were not sufficient with the columns of the other stations, the date can be used as additional data to add seasonal information.
- If the algorithms are not sufficient, a neural network of type RNN - LSTM can be explored.

Let's go for a regression algorithm

Before testing a set of regression algorithms, we added two features to the dataset, the sine and cosine of the "day" of the year, which goes between 0 and 360, taking the first of January as day 1 and so ascending to 359. The last days of the year, greater or equal to 359 are equal to zero to simplify the conversion.

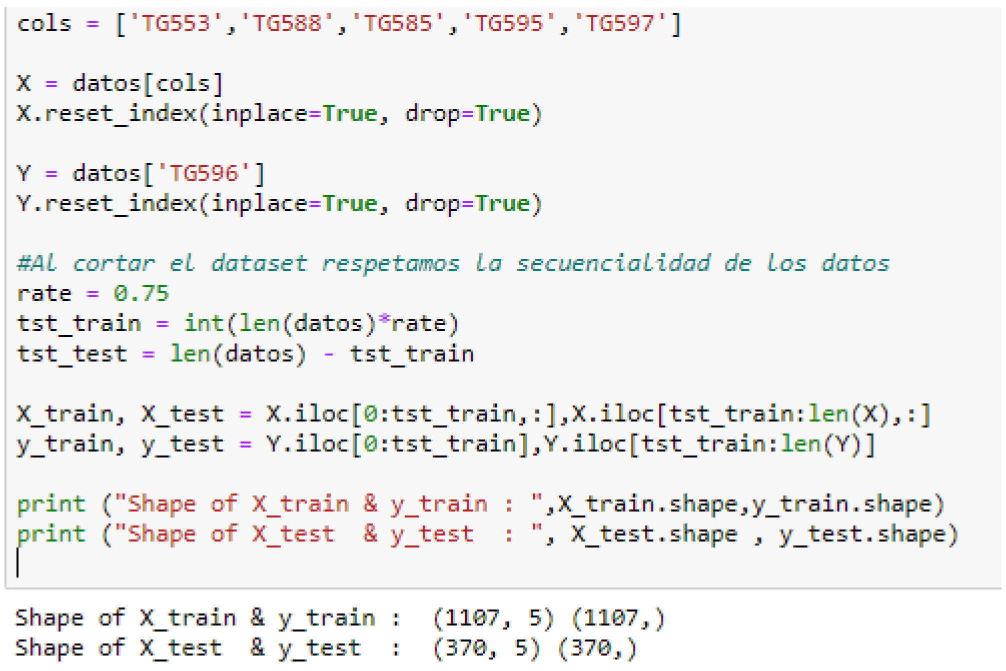
Let's remember that the algorithms of numpy for trignomoetria have their income in radians, for that reason we do the conversion to degrees before obtaining sine and cosine:

```
datos['Day'] = 0
length_months = [31,28,31,30,31,30,31,31,30,31,30,31]

for idx,row in datos.iterrows():
    day = idx.day
    if idx.month > 1:
        for m in range(idx.month-1):
            day = day + length_months[m]

    if day >= 360:
        day = 0

    datos.loc[datos.index == idx ,'Day'] = float(day)

datos['Sin'] = np.abs( np.sin(datos['Day'] * ((2*np.pi)/360) ) )
datos['Cos'] = np.abs(np.cos(datos['Day'] * ((2*np.pi)/360)) )
datos.head(10)
```

	TG553	TG585	TG588	TG595	TG596	TG597	Day	Sin	Cos
DATE									
1976-12-28	30.0	100.0	250.0	260.0	236.0	237.0	0.0	0.000000	1.000000
1976-12-29	55.0	101.0	246.0	248.0	239.0	239.0	0.0	0.000000	1.000000
1976-12-30	53.0	99.0	253.0	230.0	249.0	242.0	0.0	0.000000	1.000000
1976-12-31	70.0	105.0	248.0	229.0	241.0	236.0	0.0	0.000000	1.000000
1977-01-01	55.0	111.0	245.0	227.0	238.0	239.0	1.0	0.017452	0.999848
1977-01-02	46.0	101.0	256.0	234.0	250.0	240.0	2.0	0.034899	0.999391
1977-01-05	71.0	102.0	234.0	217.0	229.0	216.0	5.0	0.087156	0.996195

If at any time we need these features, we will have to scale the rest of the columns to leave the information in a homogeneous range and not divert the internal functioning of the algorithms or networks we use.

Prepare data entry to a regression model:



```
cols = ['TG553','TG588','TG585','TG595','TG597']

X = datos[cols]
X.reset_index(inplace=True, drop=True)

Y = datos['TG596']
Y.reset_index(inplace=True, drop=True)

#Al cortar el dataset respetamos la secuencialidad de los datos
rate = 0.75
tst_train = int(len(datos)*rate)
tst_test = len(datos) - tst_train

X_train, X_test = X.iloc[0:tst_train,:],X.iloc[tst_train:len(X),:]
y_train, y_test = Y.iloc[0:tst_train],Y.iloc[tst_train:len(Y)]

print ("Shape of X_train & y_train : ",X_train.shape,y_train.shape)
print ("Shape of X_test  & y_test  : ", X_test.shape , y_test.shape)
```

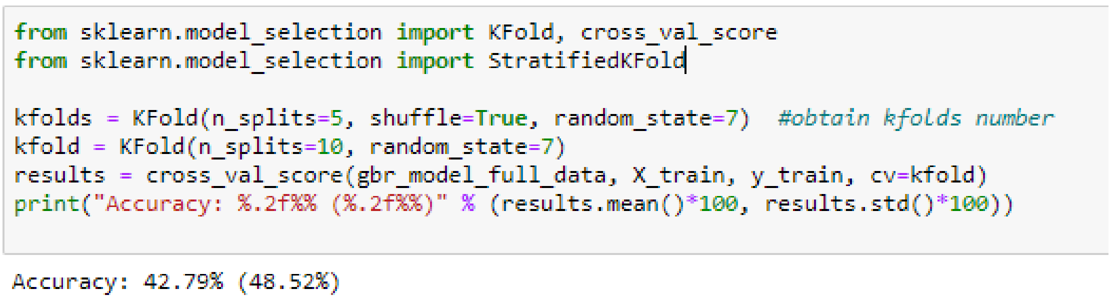
```
Shape of X_train & y_train :  (1107, 5) (1107,)
Shape of X_test  & y_test  :  (370, 5) (370,)
```

After applying GradientBoostingRegressor from sklearn.ensemble, we obtain the following results:

```
Gradiente RMSLE score on train data:
5.4432688838663195
Accuracy -->  98.16939124093663

Gradiente RMSLE score on test data:
16.675179454983343
Accuracy -->  63.10001902521414
```

This means that in a first approximation with the training data an interesting accuracy is obtained (which can be taken as a basis to start trying to improve); however, with the test data the precision of the prediction collapses: overfitting.

Let's try to apply KFold to our algorithm, we try with groups of data of 10 and 5, and in both cases the score is very low, which tells us that a priori as much precision as we get in training, when applied to the separate data set for testing, the accuracy is going to be bad.



```
from sklearn.model_selection import KFold, cross_val_score
from sklearn.model_selection import StratifiedKFold

kfolds = KFold(n_splits=5, shuffle=True, random_state=7)  #obtain kfolds number
kfold = KFold(n_splits=10, random_state=7)
results = cross_val_score(gbr_model_full_data, X_train, y_train, cv=kfold)
print("Accuracy: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

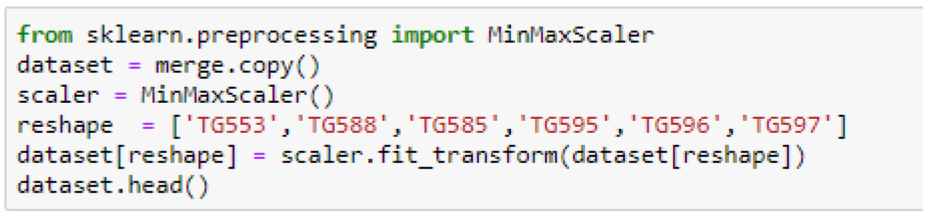
```
Accuracy: 42.79% (48.52%)
```

Well, by definitely getting the cross validation accuracy, we confirm that we're doing very badly. The algorithm itself with the dataset is trained well, but it is lost when we take it out to the testing dataset, or if we put it into production.

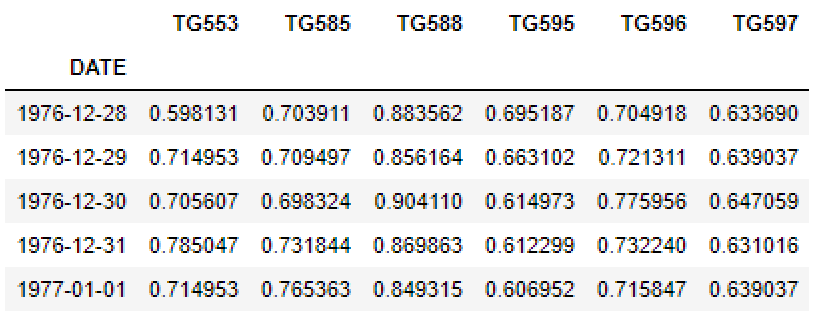
We could apply different techniques to try to improve this, but luckily there are alternative strategies and the baseline leads us to try directly with a LSTM network to see if we start, we have better results..

LSTM neuronal network

First let's scale the data using MinMaxScaler



```
from sklearn.preprocessing import MinMaxScaler
dataset = merge.copy()
scaler = MinMaxScaler()
reshape  = ['TG553','TG588','TG585','TG595','TG596','TG597']
dataset[reshape] = scaler.fit_transform(dataset[reshape])
dataset.head()
```



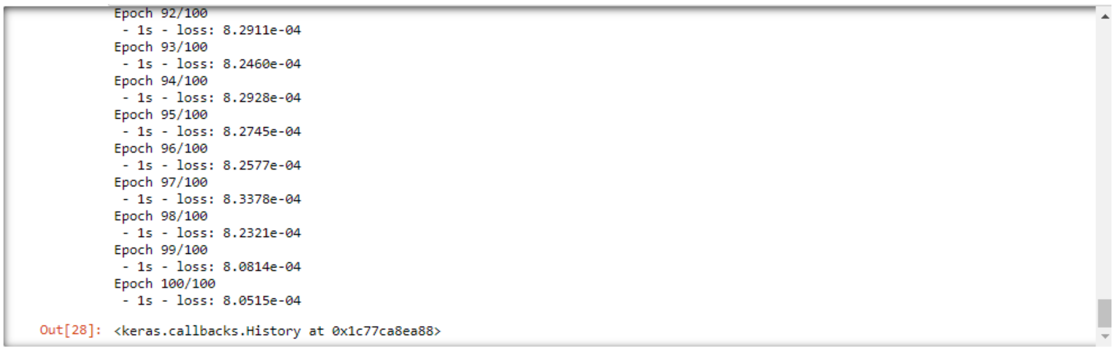
	TG553	TG585	TG588	TG595	TG596	TG597
DATE						
1976-12-28	0.598131	0.703911	0.883562	0.695187	0.704918	0.633690
1976-12-29	0.714953	0.709497	0.856164	0.663102	0.721311	0.639037
1976-12-30	0.705607	0.698324	0.904110	0.614973	0.775956	0.647059
1976-12-31	0.785047	0.731844	0.869863	0.612299	0.732240	0.631016
1977-01-01	0.714953	0.765363	0.849315	0.606952	0.715847	0.639037

First let's scale the data using MinMaxScaler:

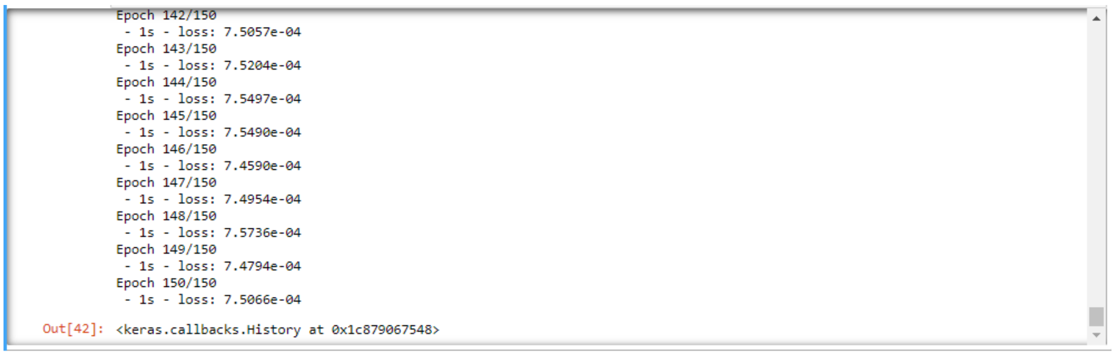
```
separador = 0.75
tst_train = int(len(dataset)*separador)
tst_test = len(dataset) - tst_train
dataset.reset_index(inplace=True, drop=True)

print (dataset.head())
print (tst_train)
train, test = dataset.loc[0:tst_train,:],dataset.loc[tst_train:len(dataset),:]

print (train.shape)
print (test.shape)
```

```
      TG553     TG585     TG588     TG595     TG596     TG597
0  0.598131  0.703911  0.883562  0.695187  0.704918  0.633690
1  0.714953  0.709497  0.856164  0.663102  0.721311  0.639037
2  0.705607  0.698324  0.904110  0.614973  0.775956  0.647059
3  0.785047  0.731844  0.869863  0.612299  0.732240  0.631016
4  0.714953  0.765363  0.849315  0.606952  0.715847  0.639037
1107
(1108, 6)
(370, 6)
```

We build a LSTM network of 4 perceptrons, with a look back window of 2 elements, we train it during 100 and 150 epochs with a batch_size of 3, obtaining:



```
        Epoch 92/100
         - 1s - loss: 8.2911e-04
        Epoch 93/100
         - 1s - loss: 8.2460e-04
        Epoch 94/100
         - 1s - loss: 8.2928e-04
        Epoch 95/100
         - 1s - loss: 8.2745e-04
        Epoch 96/100
         - 1s - loss: 8.2577e-04
        Epoch 97/100
         - 1s - loss: 8.3378e-04
        Epoch 98/100
         - 1s - loss: 8.2321e-04
        Epoch 99/100
         - 1s - loss: 8.0814e-04
        Epoch 100/100
         - 1s - loss: 8.0515e-04
Out[28]: <keras.callbacks.History at 0x1c77ca8ea88>
```

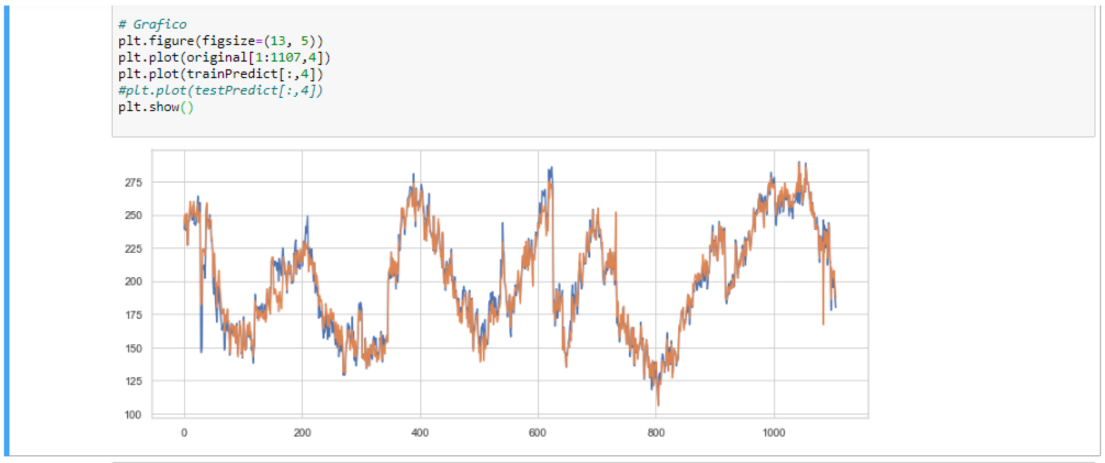


```
        Epoch 142/150
         - 1s - loss: 7.5057e-04
        Epoch 143/150
         - 1s - loss: 7.5204e-04
        Epoch 144/150
         - 1s - loss: 7.5497e-04
        Epoch 145/150
         - 1s - loss: 7.5490e-04
        Epoch 146/150
         - 1s - loss: 7.4590e-04
        Epoch 147/150
         - 1s - loss: 7.4954e-04
        Epoch 148/150
         - 1s - loss: 7.5736e-04
        Epoch 149/150
         - 1s - loss: 7.4794e-04
        Epoch 150/150
         - 1s - loss: 7.5066e-04
Out[42]: <keras.callbacks.History at 0x1c879067548>
```

Obtaining accuracy (4.71 tenths of a degree in the daily mean is a reasonable accuracy for an initial attempt):

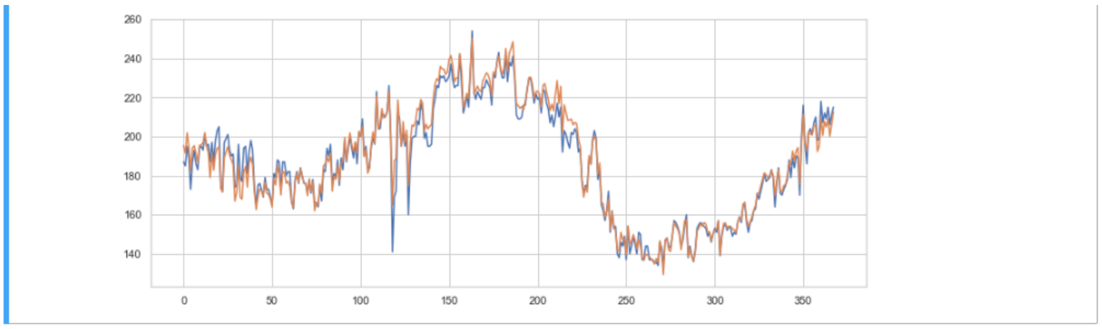
```
Train: 6.56 RMSE
Test: 4.71 RMSE
```

Training set and predictions:



```
# Grafico
plt.figure(figsize=(13, 5))
plt.plot(original[1:1107,4])
plt.plot(trainPredict[:,4])
#plt.plot(testPredict[:,4])
plt.show()
```

Testing set ans predictions:



We changed the short memory window to 5 look back records and we got values that didn't make it better, even things seem to have worked a little worse ...

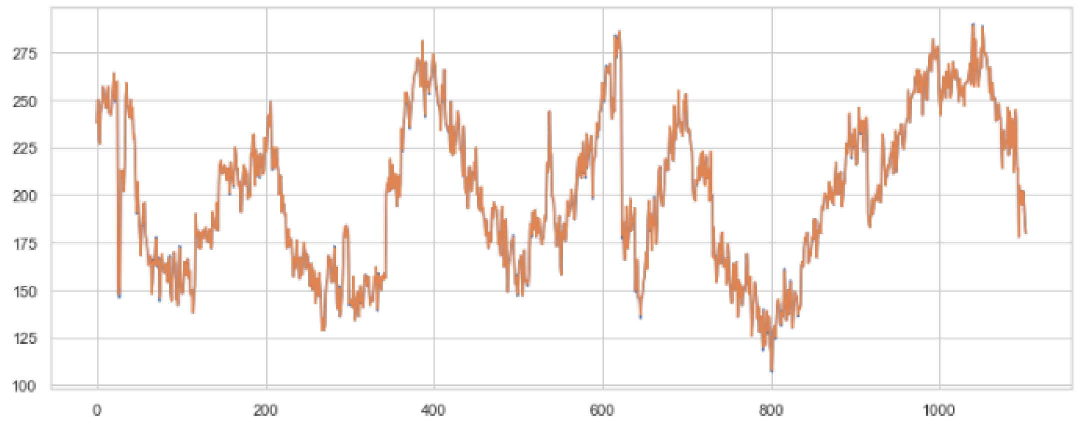
```
Train: 6.72 RMSE
Test: 5.04 RMSE
```

We modify the structure of our neural network, increasing the number of perceptrons from 4 to 6 and then yes, the accuracy changes and improves substantially (although t he training of the network is a little slower):

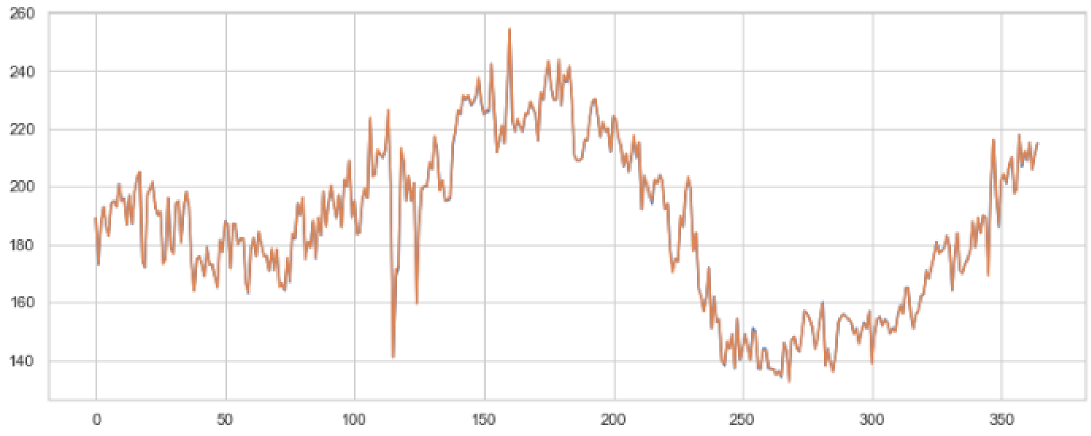
```
Train: 0.58 RMSE
Test: 0.46 RMSE
```

Now, we plot again train set and predictions :

```
# Grafico
plt.figure(figsize=(13, 5))
plt.plot(original[4:1107,4])
plt.plot(trainPredict[:,4])
#plt.plot(testPredict[:,4])
plt.show()
```



(it looks like a single line but it really is two, believe me!)



Once the definitive model has been obtained, it must be applied to the data line of the 596-Azapa weather station, indicating with a "9" in the Q_TG column, to indicate that the data was obtained artificially and not through the actual initial records.

Summary

Once obtained the fine data set to enter the predictive models, it is good to consider a strategy, have alternatives, an idea, but it is also good to make a survey between the different "line sbases" and see what preliminary results are obtained, respecting the rule of always going from the simplest to the most complex.