ensayo4

September 18, 2015

1 Análisis de los datos obtenidos

Uso de ipython para el análsis y muestra de los datos obtenidos durante la producción.Se implementa un regulador experto. Los datos analizados son del día 13 de Agosto del 2015

Los datos del experimento: * Hora de inicio: 10:30 * Hora final : 11:00 * Filamento extruido: 447cm * T : 150^oC * $V_{min}tractora$: 1.5mm/s * $V_{max}tractora$: 3.4mm/s * Los incrementos de velocidades en las reglas del sistema experto son distintas: * En los caso 3 y 5 se mantiene un incremento de +2. * En los casos 4 y 6 se reduce el incremento a -1.

In [1]:
```
#Importamos las librerías utilizadas
import numpy as np
import pandas as pd
import seaborn as sns
```

In [2]:
```
#Mostramos las versiones usadas de cada librerías
print ("Numpy v{}".format(np.__version__))
print ("Pandas v{}".format(pd.__version__))
print ("Seaborn v{}".format(sns.__version__))
```

```
Numpy v1.9.2
Pandas v0.16.2
Seaborn v0.6.0
```

In [3]:
```
#Abrimos el fichero csv con los datos de la muestra
datos = pd.read_csv('ensayo1.CSV')
```

In [4]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [5]:
```
#Almacenamos en una lista las columnas del fichero con las que vamos a trabajar
columns = ['Diametro X','Diametro Y', 'RPM TRAC']
```

In [6]:
```
#Mostramos un resumen de los datos obtenidoss
datos[columns].describe()
#datos.describe().loc['mean',['Diametro X [mm]', 'Diametro Y [mm]']]
```

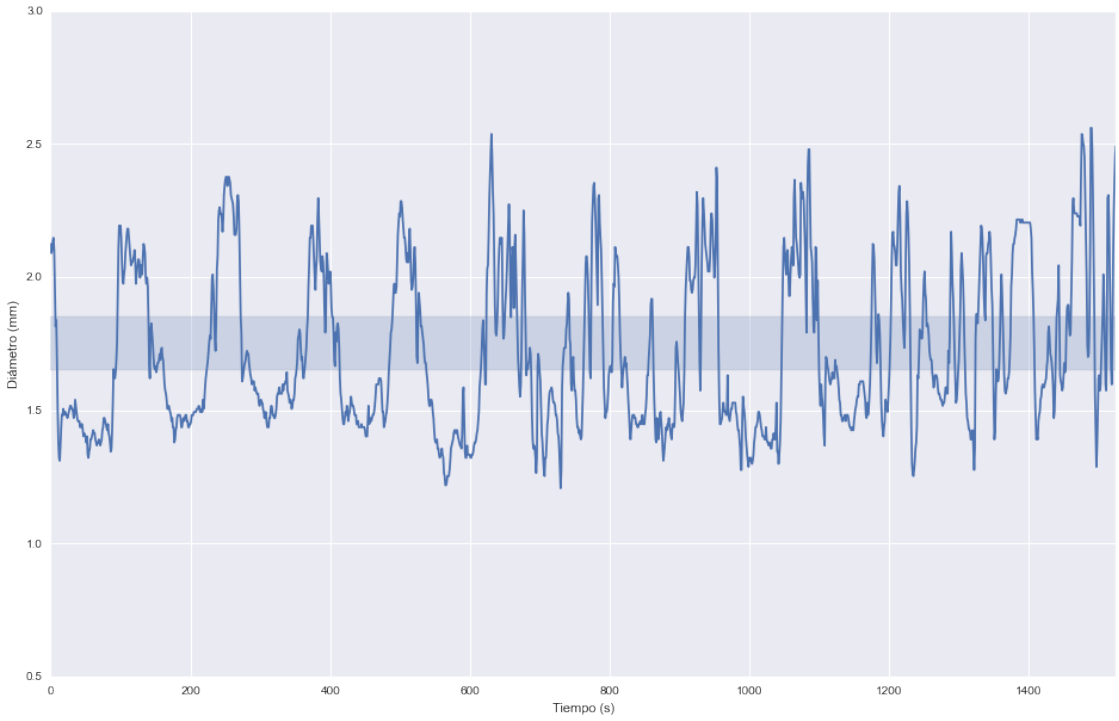
Out[6]:

	Diametro X	Diametro Y	RPM TRAC
count	1526.000000	1526.000000	1526.000000
mean	1.721607	1.707735	2.363879
std	0.299929	0.292269	0.909141
min	1.206868	1.195617	1.497500
25%	1.482145	1.471450	1.497500
50%	1.631253	1.620859	2.165000
75%	1.986820	1.942664	3.500000
max	2.560314	2.609260	3.500000

Representamos ambos diámetro y la velocidad de la tractora en la misma gráfica

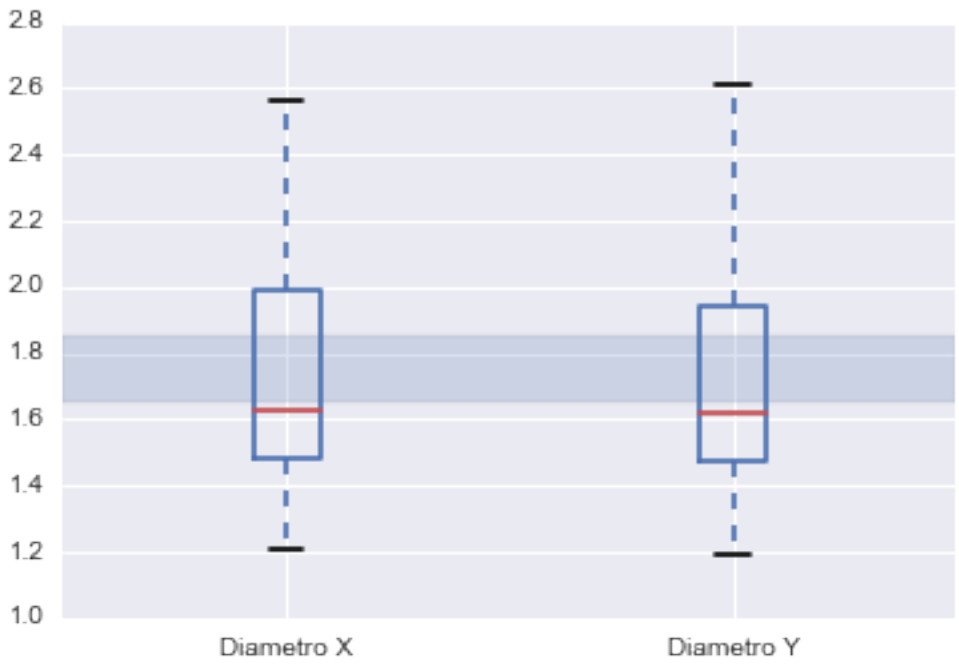
In [19]:
```
graf = datos.ix[:, "Diametro X"].plot(figsize=(16,10),ylim=(0.5,3))
graf.axhspan(1.65,1.85, alpha=0.2)
graf.set_xlabel('Tiempo (s)')
graf.set_ylabel('Diámetro (mm)')
#datos['RPM TRAC'].plot(secondary_y='RPM TRAC')
```

Out[19]: <matplotlib.text.Text at 0x9b73910>



In [8]:
```
box = datos.ix[:, "Diametro X":"Diametro Y"].boxplot(return_type='axes')
box.axhspan(1.65,1.85, alpha=0.2)
```

Out[8]: <matplotlib.patches.Polygon at 0x8accc30>



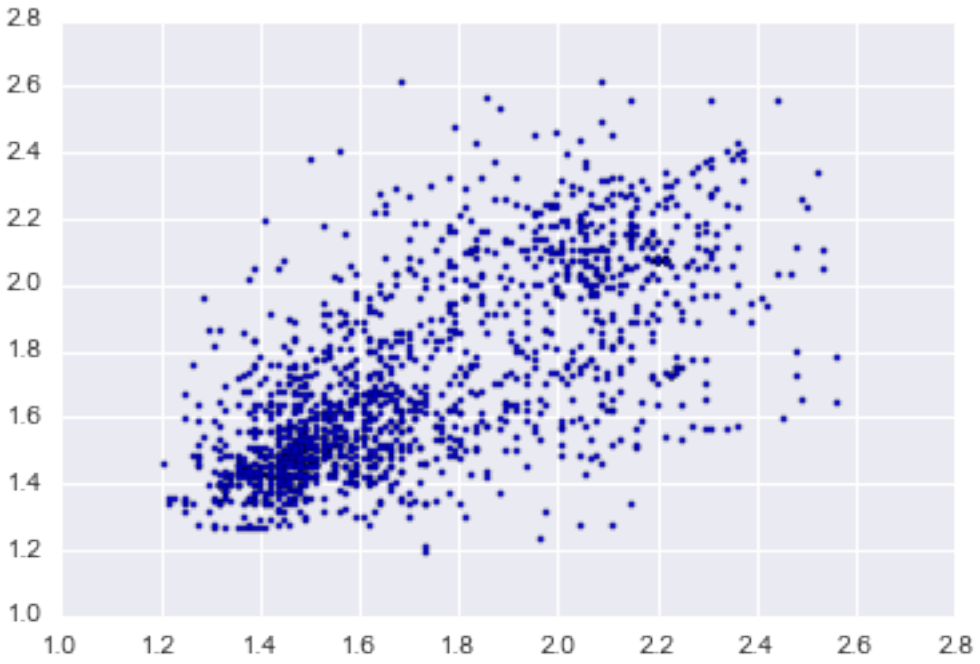
Con esta segunda aproximación se ha conseguido estabilizar los datos. Se va a tratar de bajar ese porcentaje. Como cuarta aproximación, vamos a modificar las velocidades de tracción. El rango de velocidades propuesto es de 1.5 a 5.3, manteniendo los incrementos del sistema experto como en el actual ensayo.

In [ ]:

Comparativa de Diametro X frente a Diametro Y para ver el ratio del filamento

```
In [9]: plt.scatter(x=datos['Diametro X'], y=datos['Diametro Y'], marker='.')
```

```
Out[9]: <matplotlib.collections.PathCollection at 0x8aacb70>
```



2 Filtrado de datos

Las muestras tomadas $d_x >= 0.9$ or $d_y >= 0.9$ las asumimos como error del sensor, por ello las filtramos de las muestras tomadas.

```
In [10]: datos_filtrados = datos[(datos['Diametro X'] >= 0.9) & (datos['Diametro Y'] >= 0.9)]
```

```
In [11]: #datos_filtrados.ix[:, "Diametro X":"Diametro Y"].boxplot(return_type='axes')
```

2.1 Representación de X/Y

```
In [12]: plt.scatter(x=datos_filtrados['Diametro X'], y=datos_filtrados['Diametro Y'], marker='.')
```

```
Out[12]: <matplotlib.collections.PathCollection at 0x8b39b70>
```

3 Analizamos datos del ratio

```
In [13]: ratio = datos_filtrados['Diametro X']/datos_filtrados['Diametro Y']
         ratio.describe()
```

```
Out[13]: count    1526.000000
         mean        1.016194
         std         0.135635
         min         0.632548
         25%         0.940269
         50%         0.999491
         75%         1.077269
         max         1.655858
         dtype: float64
```

```
In [14]: rolling_mean = pd.rolling_mean(ratio, 50)
         rolling_std = pd.rolling_std(ratio, 50)
         rolling_mean.plot(figsize=(12,6))
         # plt.fill_between(ratio, y1=rolling_mean+rolling_std, y2=rolling_mean-rolling_std, alpha=0.5)
         ratio.plot(figsize=(12,6), alpha=0.6, ylim=(0.5,1.5))
```

```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x8b624b0>
```

4 Límites de calidad

Calculamos el número de veces que traspasamos unos límites de calidad. $Th^+ = 1.85$ and $Th^- = 1.65$

```
In [15]: Th_u = 1.85
         Th_d = 1.65
```

```
In [16]: data_violations = datos[(datos['Diametro X'] > Th_u) | (datos['Diametro X'] < Th_d) |
                               (datos['Diametro Y'] > Th_u) | (datos['Diametro Y'] < Th_d)]
```

```
In [17]: data_violations.describe()
```

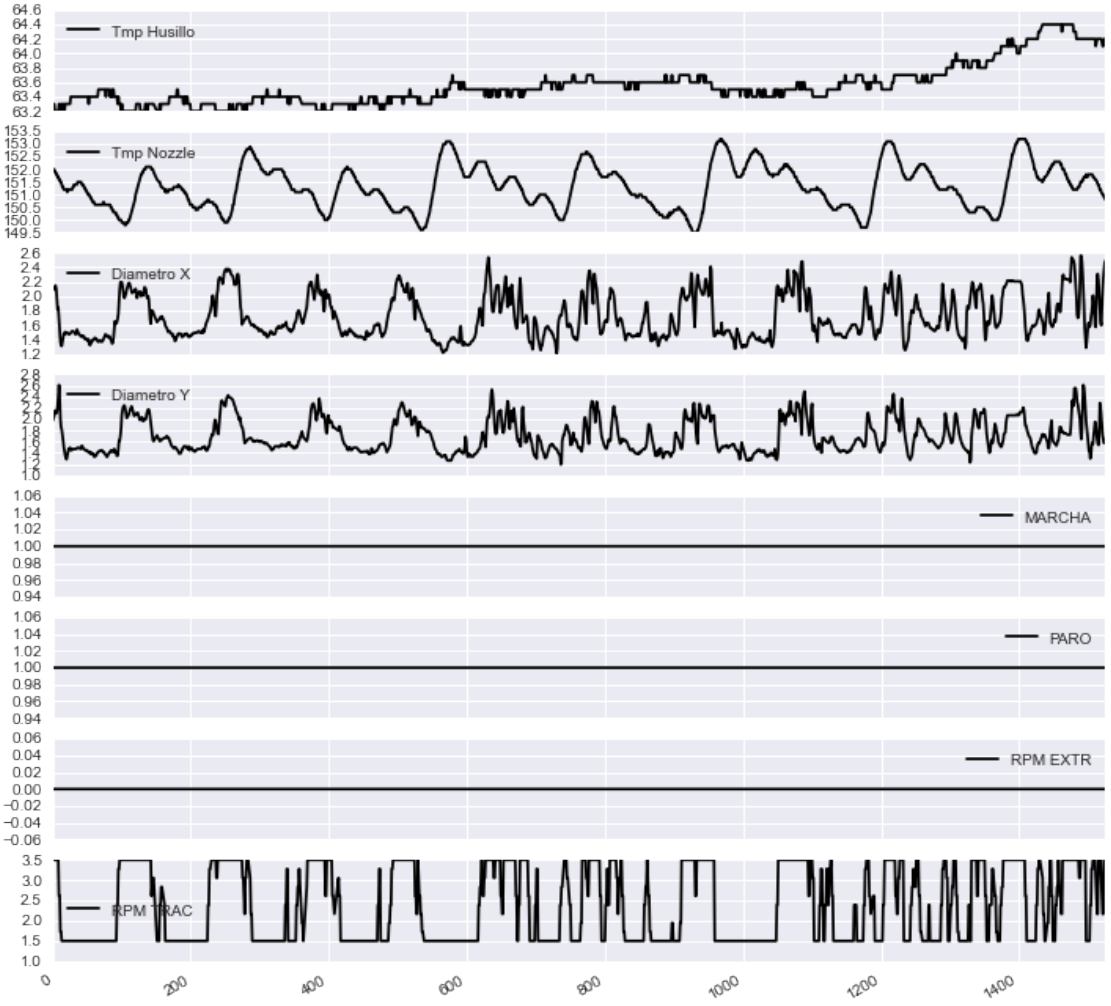
Out[17]:

	Tmp Husillo	Tmp Nozzle	Diametro X	Diametro Y	MARCHA	PARO
count	1469.000000	1469.000000	1469.000000	1469.000000	1469	1469
mean	63.554323	151.313070	1.721209	1.706638	1	1
std	0.279066	0.864951	0.305449	0.297587	0	0
min	63.200000	149.500000	1.206868	1.195617	True	True
25%	63.400000	150.600000	1.470675	1.471450	1	1
50%	63.500000	151.300000	1.619783	1.609366	1	1
75%	63.600000	151.900000	1.998289	1.954157	1	1
max	64.400000	153.200000	2.560314	2.609260	True	True

	RPM EXTR	RPM TRAC
count	1469	1469.000000
mean	0	2.356450
std	0	0.913973
min	0	1.497500
25%	0	1.497500
50%	0	1.942500
75%	0	3.500000
max	0	3.500000

```
In [18]: data_violations.plot(subplots=True, figsize=(12,12))
```

Out[18]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x08BFC270>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x08CD3970>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x08CFCD70>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x08D804B0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x08DAA790>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x08DCAB90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x08DF2DD0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x08DFD7D0>], dtype=object)
```



In [ ]: