Juan Manuel Sanchez Pareja -1010107723

```
# Ciclo que itera los modelos, los estilos de linea y los colores
# de los parametros de la funcion zip
for model, style, color in zip(models, linestyles, colors):
        # print "Modelo:",model
        # print "Coeffs:",model.coeffs

# Se define la grafica y los estilos de la grafica
        plt.plot(mx, model(mx), linestyle=style, linewidth=2, c=color)

#Poner la etiqueta con el orden del modelo en la esquina superior izquierda
    plt.legend(["d=%i" % m.order for m in models], loc="upper left")

# Ajusta la escala de los ejes del grafico
  plt.autoscale(tight=True)
  plt.ylim(ymin=0)
  if ymax:
    plt.ylim(ymax=ymax)
  if xmin:
    plt.xlim(xmin=xmin)

# Grafica las celdas que van detras del grafico
  plt.grid(True, linestyle='-', color='0.75')

# Guarda el grafico
  plt.savefig(fname)

# Primera mirada a los datos
```

```
# ------------------------------------------------------------

# Grafica el primer modelo con los primeros datos
# lo guarda en la carpeta "chart"
plot_models(x, y, None, os.path.join(CHART_DIR, "1400_01_01.png"))

# Crea y dibuja los modelos de datos
# ------------------------------------------------------------

# La funcion polyfit se descompone y se asigna a las variables
# que fp, res, rank, sv, rcond
fp1, res1, rank1, sv1, rcond1 = np.polyfit(x, y, 1, full=True)
print("Parámetros del modelo fp1: %s" % fp1)
print("Error del modelo fp1:", res1)
f1 = sp.poly1d(fp1)

fp2, res2, rank2, sv2, rcond2 = np.polyfit(x, y, 2, full=True)
print("Parámetros del modelo fp2: %s" % fp2)
print("Error del modelo fp2:", res2)
f2 = sp.poly1d(fp2)


# Aqui se estan definiendo las funciones de grado 3, 10, 100
f3 = sp.poly1d(np.polyfit(x, y, 3))
f10 = sp.poly1d(np.polyfit(x, y, 10))
f100 = sp.poly1d(np.polyfit(x, y, 100))

# Se grafican los modelos y se guardan en la carpeta "chart"
# ------------------------------------------------------------
```

```
plot_models(x, y, [f1], os.path.join(CHART_DIR, "1400_01_02.png"))
plot_models(x, y, [f1, f2], os.path.join(CHART_DIR, "1400_01_03.png"))
plot_models(
    x, y, [f1, f2, f3, f10, f100], os.path.join(CHART_DIR,
                                    "1400_01_04.png"))

# Ajusta y dibuja un modelo utilizando el conocimiento del punto
# de inflexión
# -------------------------------------------------------------------

# Se busca el punto de inflexion en el cual la funcion pasa de un
# tipo de concavidad a otra
inflexion = 3.5 * 7 * 24
xa = x[:int(inflexion)]
ya = y[:int(inflexion)]
xb = x[int(inflexion):]
yb = y[int(inflexion):]

# Se definen dos líneas rectas donde se muestra una funcion lineal
# que separa los datos desde el punto de inflexion
# -------------------------------------------------------------------
fa = sp.poly1d(np.polyfit(xa, ya, 1))
fb = sp.poly1d(np.polyfit(xb, yb, 1))

# Se grafica el modelo basado en el punto de inflexión
# -------------------------------------------------------------------
plot_models(x, y, [fa, fb], os.path.join(CHART_DIR, "1400_01_05.png"))

# Se define la función de error de acuerdo a las coordenadas
```

```
# ---------------------------------------------------------------
def error(f, x, y):
    return np.sum((f(x) - y) ** 2)

# Se imprimen los errores para el conjunto completo
# ---------------------------------------------------------------
print("Errores para el conjunto completo de datos:")
for f in [f1, f2, f3, f10, f100]:
    print("Error d=%i: %f" % (f.order, error(f, x, y)))

# Se imprimen los errores solamente después del punto de inflexión
print("Errores solamente después del punto de inflexión")
for f in [f1, f2, f3, f10, f100]:
    print("Error d=%i: %f" % (f.order, error(f, xb, yb)))

# Se imprimen los errores de inflexión
print("Error de inflexión=%f" % (error(fa, xa, ya) + error(fb, xb, yb)))

# Se extrapola de modo que se proyecten respuestas en el futuro
# ---------------------------------------------------------------

# Se extraen las hipótesis de valores futuros de los datos analizados
plot_models(
    x, y, [f1, f2, f3, f10, f100],
    os.path.join(CHART_DIR, "1400_01_06.png"),
    mx=np.linspace(0 * 7 * 24, 6 * 7 * 24, 100),
    ymax=10000, xmin=0 * 7 * 24)
```