In [1]:

```
%matplotlib inline
```

Selección de características

1.- Librerias

In [2]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import re
from scipy import stats,ndimage

#Para convertir los datos que son categóricos
import sklearn.preprocessing as pp

import dateutil
#Hay que istalar esta librería que hace el parseo del user agent
#pip install pyyaml ua-parser user-agents
from user_agents import parse

#Base maps -> mirar como instalarlo en la bibliografía al final del documen
to
#http://gnperdue.github.io/yak-shaving/osx/python/matplotlib/2014/05/01/bas
emap-toolkit.html
from mpl_toolkits.basemap import Basemap


#Para pintar gráficos vistosos usamos seaborn:
import seaborn as sns

#y creamos la paleta:
sns.set_palette("deep", desat=.6)
sns.set_context(rc={"figure.figsize": (8, 4)})
```

2.- Descripcion de los datos

DESPUES DE ANONIMIZAR Y SELECCIONAR ÚNICAMENTE LAS VARIABLES QUE QUEREMOS

num_columna	Nombre	Descripción	Variable
1	ciudad	ciuda de origen del usuario	discreta
2	email_server	servidor de email del usuario	discreta
3	**edad**	edad del usuario (variable objetivo)	discreta
4	**genero**	genero del usuario (variable objetivo)	discreta
6	hora_visita	hora en que el usuario hace la visita	discreta
7	is_weekend	fin de semana	discreta
8	nombre_final	nombre del usuario	discreta
9	os	sistema operativo	discreta
10	pais	pais en el user agent	discreta
11	rango horario	momento del día en que se conecta el usuario	discreta
12	time_zone	zona horaria del usuario	discreta
13	ua_browser_family	familia del navegador en el user agent	discreta
14	ua_device	dispositivo que utiliza el usuario segun user agent	discreta
15	ua_device_family	familia del dispositivo en el user agent	discreta
16	ua_is_bot	si es un robot	discreta
17	ua_is_movile	si es un movil	discreta
19	ua_is_pc	si es un pc	discreta
20	ua_is_tablet	si es una tablet	discreta
21	ua_is_tounch_capable	si es táctil	discreta
22	ua_os_family	familia sistema operativo	discreta
23	weekday	dia de la semana	discreta
24	id_hotsposts	local	discreta

```
-Faltaría saber si se ha conectado con facebook, google o email (deb
ería hacerlo en la recolección de variables), así como rellenar los
nulos con un valor ("vacio")
-También faltaría la categoría del local en que se ha conectado y ha
cer algo con las provincias.
```

3.- Carga de los datos

Cargamos los datos que hemos limpiado anteriormente y guardado en un csv para cargarlos más fácilmente). Al final del ejercicio habría que integrarlo todo en un único proceso para su uso.

In [3]:

```
df = pd.read_csv('../csv/datos_explorados.csv')

#borro la columna unnamed
df.drop('Unnamed: 0', axis=1,inplace=True)

print df.columns
```

```
Index([u'ciudad', u'email_server', u'edad', u'genero', u'hora_visita', u'id
ioma', u'is_weekend', u'nombre_final', u'os', u'pais', u'rango_horario',
u'timezone', u'ua_browser_family', u'ua_device', u'ua_device_family', u'u
a_is_bot', u'ua_is_movile', u'ua_is_pc', u'ua_is_tablet', u'ua_is_tounch_ca
pable', u'ua_os_family', u'weekday', u'id_hotspots'], dtype='object')
```

In [4]:

```
#vamos a poner el tipo de los datos:

df.ciudad = df.ciudad.astype('category')
df.email_server = df.email_server.astype('category')
df.edad = df.edad.astype('category')
df.genero = df.genero.astype('category')
df.hora_visita = df.hora_visita.astype('category')
df.idioma = df.idioma.astype('category')
df.is_weekend = df.is_weekend.astype('category')
df.nombre_final = df.nombre_final.astype('category')
df.os = df.os.astype('category')
df.pais = df.pais.astype('category')
df.rango_horario = df.rango_horario.astype('category')
df.timezone = df.timezone.astype('category')
df.ua_browser_family = df.ua_browser_family.astype('category')
df.ua_device = df.ua_device.astype('category')
df.ua_device_family = df.ua_device_family.astype('category')
df.ua_is_bot = df.ua_is_bot.astype('category')
df.ua_is_movile = df.ua_is_movile.astype('category')
df.ua_is_pc = df.ua_is_pc.astype('category')
df.ua_is_tablet = df.ua_is_tablet.astype('category')
df.ua_is_tounch_capable = df.ua_is_tounch_capable.astype('category')
df.ua_os_family = df.ua_os_family.astype('category')
df.weekday = df.weekday.astype('category')
df.id_hotspots = df.id_hotspots.astype('category')
```

4.- FEATURE SELECTION - SCIKIT EXTRA TREE CLASIFIER

In [5]:

```
df.columns
```

Out[5]:

```
Index([u'ciudad', u'email_server', u'edad', u'genero', u'hora_visita', u'id
ioma', u'is_weekend', u'nombre_final', u'os', u'pais', u'rango_horario',
u'timezone', u'ua_browser_family', u'ua_device', u'ua_device_family', u'u
a_is_bot', u'ua_is_movile', u'ua_is_pc', u'ua_is_tablet', u'ua_is_tounch_ca
pable', u'ua_os_family', u'weekday', u'id_hotspots'], dtype='object')
```

In [6]:

```
#Hacemos un label_encoder con las categoricas de tipo string
from sklearn import preprocessing
df_features = pd.DataFrame()

#ciudad
le_ciudad = preprocessing.LabelEncoder()
le_ciudad.fit(df.ciudad.unique())

#email_server
le_email_server = preprocessing.LabelEncoder()
le_email_server.fit(df.email_server.unique())

#hora_visita --> ya es numérico

#idioma
le_idioma = preprocessing.LabelEncoder()
le_idioma.fit(df.idioma.unique())

#is_weeked
le_is_weekend = preprocessing.LabelEncoder()
le_is_weekend.fit(df.is_weekend.unique())

#os
le_os = preprocessing.LabelEncoder()
le_os.fit(df.os.unique())

#pais
le_pais = preprocessing.LabelEncoder()
le_pais.fit(df.pais.unique())

#rango_horario
le_rango_horario = preprocessing.LabelEncoder()
le_rango_horario.fit(df.rango_horario.unique())

#time_zone
le_time_zone = preprocessing.LabelEncoder()
le_time_zone.fit(df.timezone.unique())


#ua_browser_family
le_browser_family = preprocessing.LabelEncoder()
le_browser_family.fit(df.ua_browser_family.unique())

#ua_device

le_device = preprocessing.LabelEncoder()
le_device.fit(df.ua_device.unique())

#ua_device_family
le_device_family = preprocessing.LabelEncoder()
le_device_family.fit(df.ua_device_family.unique())

#ua_is_bot <- siempre false, no lo usamos

#ua_is_movile
```

```
#ua_is_pc
#ua_is_tablet
#ua_is_tounch_capable

#ua_os_family
le_os_family = preprocessing.LabelEncoder()
le_os_family.fit(df.ua_os_family.unique())

#weekday' <- ya es numerico

df_features["ciudad"] = le_ciudad.transform(df.ciudad)
df_features["email_server"] = le_email_server.transform(df.email_server)
df_features["hora_visita"] = df.hora_visita
df_features["idioma"] = le_idioma.transform(df.idioma)
df_features["is_weekend"] = [1 if x else 0 for x in df.is_weekend]
df_features["os"] = le_os.transform(df.os)
df_features["pais"] = le_pais.transform(df.pais)
df_features["rango_horario"] = le_rango_horario.transform(df.rango_horario)
df_features["time_zone"] = le_time_zone.transform(df.timezone)
df_features["browser_family"] = le_browser_family.transform(df.ua_browser_f
amily)
df_features["device"] = le_device.transform(df.ua_device)
df_features["device_family"] = le_device_family.transform(df.ua_device_fami
ly)

df_features["is_movile"] = [1 if x else 0 for x in df.ua_is_movile]
df_features["is_pc"] = [1 if x else 0 for x in df.ua_is_pc]
df_features["is_tablet"] = [1 if x else 0 for x in df.ua_is_tablet]
df_features["is_tounch_capable"] = [1 if x else 0 for x in df.ua_is_tounc
h_capable]

df_features["os_family"] = le_os_family.transform(df.ua_os_family)
df_features["weekday"] = df.weekday

df_features["id_hotspots"] = df.id_hotspots

print df_features.head()
```

```
   ciudad  email_server hora_visita  idioma  is_weekend  os  pais  \
0     347           331          20      38           0  15    49
1     347           217          20      26           1  15    33
2     347           217          23       8           1  15    20
3     347           217          21       8           1   0    26
4     184           217          19       9           0  15    17

   rango_horario  time_zone  browser_family  device  device_family  is_movi
le  \
0              2         20               5      44             64
1
1              2         20               1     145            441
1
2              4         20               5     142            409
1
3              4         20               1     142            374
1
4              2         20              14     196            523
1

   is_pc  is_tablet  is_tounch_capable  os_family weekday id_hotspots
0      0          0                  1          0       2           2
1      0          0                  1          0       5           2
2      0          0                  1          0       6           2
3      0          0                  1          0       6           2
4      0          0                  1         17       3           2
```

In [7]:

```
#Guardamos los resultados para usarlos más tarde
from sklearn.externals import joblib

joblib.dump(le_ciudad, 'models/le_ciudad.pkl')
joblib.dump(le_email_server, 'models/le_email_server.pkl')
joblib.dump(le_idioma, 'models/le_idioma.pkl')
joblib.dump(le_os, 'models/le_os.pkl')
joblib.dump(le_pais, 'models/le_pais.pkl')
joblib.dump(le_rango_horario, 'models/le_rango_horario.pkl')
joblib.dump(le_time_zone, 'models/le_time_zone.pkl')
joblib.dump(le_browser_family,'models/le_browser_family.pkl')
joblib.dump(le_rango_horario, 'models/le_rango_horario.pkl')
joblib.dump(le_device, 'models/le_device.pkl')
joblib.dump(le_device_family, 'models/le_device_family.pkl')
joblib.dump(le_os_family, 'models/le_os_family.pkl')
```

Out[7]:

```
['models/le_os_family.pkl', 'models/le_os_family.pkl_01.npy']
```

5.- Correlaciones

In [8]:

```
corrmat = df_features.corr()
corrmat
```

Out[8]:

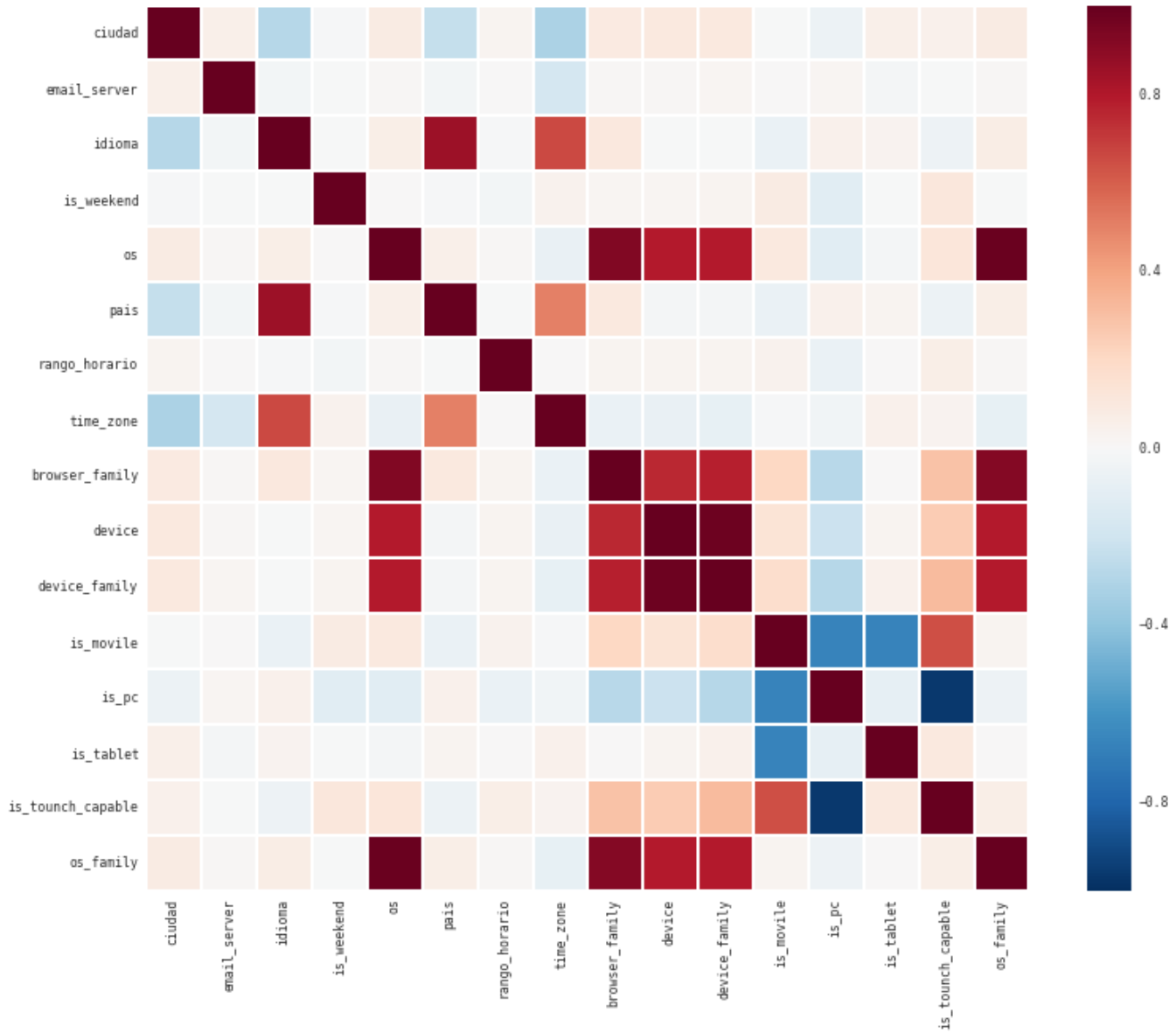
	ciudad	email_server	idioma	is_weekend	os	pais	ran
ciudad	1.000000	0.056611	-0.285667	-0.012336	0.080512	-0.237852	0.0
email_server	0.056611	1.000000	-0.025931	-0.001945	0.010108	-0.023916	0.0
idioma	-0.285667	-0.025931	1.000000	-0.005247	0.066957	0.852089	-0.0
is_weekend	-0.012336	-0.001945	-0.005247	1.000000	0.004263	-0.013841	-0.0
os	0.080512	0.010108	0.066957	0.004263	1.000000	0.060883	0.0
pais	-0.237852	-0.023916	0.852089	-0.013841	0.060883	1.000000	-0.0
rango_horario	0.028332	0.000898	-0.008358	-0.029117	0.013330	-0.003405	1.0
time_zone	-0.319447	-0.174910	0.662618	0.039243	-0.074762	0.503814	0.0
browser_family	0.086622	0.014459	0.103185	0.022269	0.922260	0.097446	0.0
device	0.095339	0.015549	-0.004842	0.020215	0.793192	-0.019455	0.0
device_family	0.099197	0.015993	-0.003131	0.029538	0.796857	-0.018535	0.0
is_movile	-0.002599	0.000082	-0.064172	0.084206	0.099302	-0.063653	0.0
is_pc	-0.055266	0.017627	0.051960	-0.113361	-0.117145	0.054280	-0.0
is_tablet	0.058745	-0.016300	0.033427	-0.000034	-0.019019	0.030205	0.0
is_tounch_capable	0.053406	-0.004646	-0.050163	0.112507	0.123124	-0.055678	0.0
os_family	0.084737	0.013701	0.071780	-0.002614	0.987061	0.066548	0.0

In [9]:

```
sns.set(context="paper", font="monospace")

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(13, 10))

# Draw the heatmap using seaborn
sns.heatmap(corrmat, vmax=1, square=True)
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x105fae810>
```



In [10]:

```
#Comprobamos la correlacion entre os y os_family y device y device_family
stats.pearsonr(df_features.device, df_features.device_family)
```

Out[10]:

```
(0.9826316367702147, 0.0)
```

In [11]:

```
print "Varianza device: " , ndimage.variance(np.array(df_features.device))
print "Variation device: " , stats.variation(df_features.device)
print
print "Varianza device_family:", ndimage.variance(np.array(df_features.devi
ce_family))
print "Variation device_family: " , stats.variation(df_features.device_fami
ly)
```

```
Varianza device:  2628.16823655
Variation device:  0.327582303992

Varianza device_family: 26164.611414
Variation device_family:  0.409143033991
```

In [12]:

```
#Comprobamos la correlacion entre os y os_family y device y device_family
stats.pearsonr(df_features.os, df_features.os_family)
```

Out[12]:

```
(0.98706087541567367, 0.0)
```

In [13]:

```
print "Varianza os: " ,ndimage.variance(np.array(df_features.os))
print "Variation os: " , stats.variation(df_features.os)
print
print "Varianza os_family:", ndimage.variance(np.array(df_features.os_famil
y))
print "Variation os_family: " , stats.variation(df_features.os_family)
```

```
Varianza os:  36.7379405628
Variation os:  0.801689631739

Varianza os_family: 62.9561993769
Variation os_family:  0.787413536339
```

In [14]:

```
#Comprobamos la correlacion entre is_touch_capaple y is_pc
stats.pearsonr(df_features.is_tounch_capable, df_features.is_pc)
```

Out[14]:

```
(-0.96361765934288923, 0.0)
```

In [15]:

```
print "Varianza is_touch_capable: " ,ndimage.variance(np.array(df_feature
s.is_tounch_capable))
print "Variation is_touch_capable: " , stats.variation(df_features.is_tounc
h_capable)
print
print "Varianza is_pc:", ndimage.variance(np.array(df_features.is_pc))
print "Variation is_pc: " , stats.variation(np.array(df_features.is_pc))
```

```
Varianza is_touch_capable:  0.0921767622028
Variation is_touch_capable:  0.338366771877

Varianza is_pc: 0.0889614307625
Variation is_pc:  3.02180626965
```

Vamos a eliminar aquellas con la varianza más grande, ya que variables con mayir varianza estan asociadas al overfitting en los modelos. Quitamos **device**, **os_family** y **is_pc** porque redundan con **device** y **os**. Hay tambien una alta correlación etre **os** y **browser_family**, pero vamos a dejarlas porque son menores del 95% Tambien quito **ciudad** porque esta poco limpiay mete mucho ruido

In [16]:

```
#Comprobamos la correlacion entre os y browser_family
stats.pearsonr(df_features.os, df_features.browser_family)
```

Out[16]:

```
(0.92225983815875234, 0.0)
```

In [17]:

```
df_features.drop(['os_family','device','ciudad', 'is_pc'], axis=1,inplace=T
rue)
```

6.- División del dataset

In [18]:

```
#vamos a dividir en train y test para genero y para edad

X_train_genero = df_features[pd.notnull(df.genero)]
X_test_genero = df_features[pd.isnull(df.genero)]
y_train_genero = df[pd.notnull(df.genero)].genero.values


X_train_edad = df_features[pd.notnull(df.edad)]
X_test_edad = df_features[pd.isnull(df.edad)]
y_train_edad =  df[pd.notnull(df.edad)].edad.values
```

7.- FEATURE SELECTION - Varianza

In [19]:

```
from sklearn.feature_selection import VarianceThreshold
sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
X_sel_new = sel.fit_transform(df_features)

print df_features.shape
print sel.get_support()
print X_sel_new.shape

df_features_varianza = df_features[df_features.columns[sel.get_support()]]
print df_features_varianza.columns
```

```
(26078, 15)
[ True  True  True  True  True  True  True  True  True  True False False
 False  True  True]
(26078, 12)
Index([u'email_server', u'hora_visita', u'idioma', u'is_weekend', u'os',
u'pais', u'rango_horario', u'time_zone', u'browser_family', u'device_famil
y', u'weekday', u'id_hotspots'], dtype='object')
```

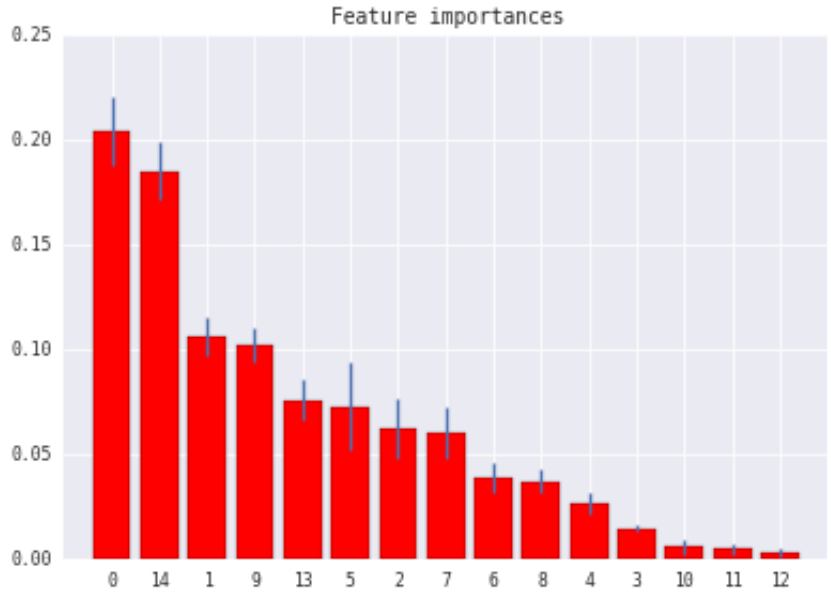
8.- FEATURE SELECTION - EXTRATREE CLASIFIER

Primero para el género

In [20]:

```
from sklearn.ensemble import ExtraTreesClassifier

X = X_train_genero
y = y_train_genero
print X.shape

clf = ExtraTreesClassifier()
X_new = clf.fit(X, y).transform(X)

print clf.feature_importances_

print X_new.shape


import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import make_classification
from sklearn.ensemble import ExtraTreesClassifier

num_features = len(X_train_genero.columns)
importances = clf.feature_importances_
std = np.std([tree.feature_importances_ for tree in clf.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]


# Print the feature ranking
print("Feature ranking:")

for f in range(num_features):
    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indice
s[f]]))



# Plot the feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(num_features), importances[indices],color="r", yerr=std[indic
es], align="center")
plt.xticks(range(num_features), indices)
plt.xlim([-1, num_features])
plt.show()
```

```
(25298, 15)
[ 0.20364268  0.10628973  0.06242492  0.0147678   0.02709537  0.07291329
  0.03903333  0.06020804  0.03728952  0.10209669  0.00594778  0.00483585
  0.00295435  0.07565157  0.1848491 ]
(25298, 6)
Feature ranking:
1. feature 0 (0.203643)
2. feature 14 (0.184849)
3. feature 1 (0.106290)
4. feature 9 (0.102097)
5. feature 13 (0.075652)
6. feature 5 (0.072913)
7. feature 2 (0.062425)
8. feature 7 (0.060208)
9. feature 6 (0.039033)
10. feature 8 (0.037290)
11. feature 4 (0.027095)
12. feature 3 (0.014768)
13. feature 10 (0.005948)
14. feature 11 (0.004836)
15. feature 12 (0.002954)
```

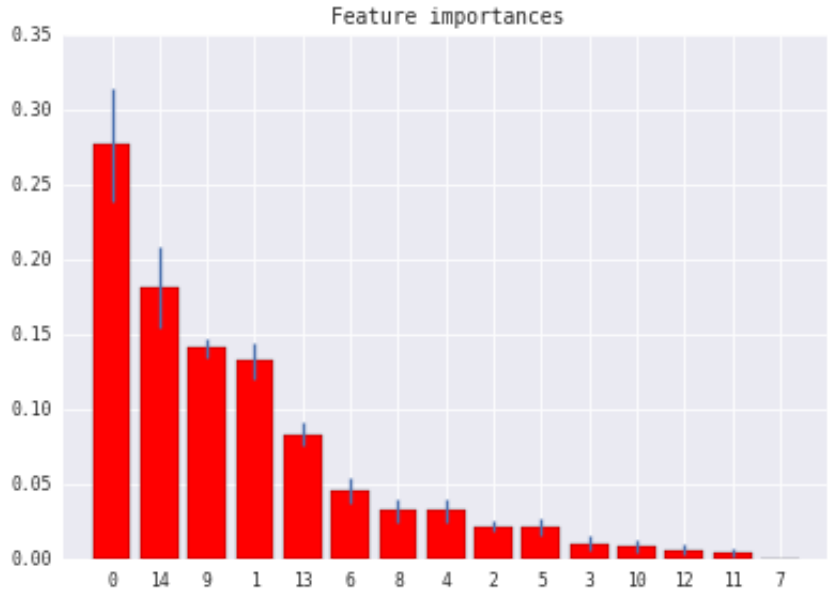


Ahora para la edad

In [21]:

```
from sklearn.ensemble import ExtraTreesClassifier
X = X_train_edad
y = y_train_edad
print X.shape

num_features = len(X_train_edad.columns)

clf = ExtraTreesClassifier()
X_new = clf.fit(X, y).transform(X)

print clf.feature_importances_

print X_new.shape


import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import make_classification
from sklearn.ensemble import ExtraTreesClassifier


importances = clf.feature_importances_
std = np.std([tree.feature_importances_ for tree in clf.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]


# Print the feature ranking
print("Feature ranking:")

for f in range(num_features):
    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indice
s[f]]))



# Plot the feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(num_features), importances[indices],color="r", yerr=std[indic
es], align="center")
plt.xticks(range(num_features), indices)
plt.xlim([-1, num_features])
plt.show()
```

```
(10895, 15)
[ 0.27630659  0.13225193  0.02203854  0.01087687  0.03247439  0.02108626
  0.0463847   0.          0.03263802  0.14083687  0.00888745  0.00472774
  0.00657072  0.08336786  0.18155206]
(10895, 5)
Feature ranking:
1. feature 0 (0.276307)
2. feature 14 (0.181552)
3. feature 9 (0.140837)
4. feature 1 (0.132252)
5. feature 13 (0.083368)
6. feature 6 (0.046385)
7. feature 8 (0.032638)
8. feature 4 (0.032474)
9. feature 2 (0.022039)
10. feature 5 (0.021086)
11. feature 3 (0.010877)
12. feature 10 (0.008887)
13. feature 12 (0.006571)
14. feature 11 (0.004728)
15. feature 7 (0.000000)
```



9.- FEATURE SELECTION - Estudio univariable (chi2, Anova (f_clasif) y Kbest)

In [22]:

```
from sklearn.feature_selection import SelectKBest, chi2, f_classif
```

Género

In [23]:

```
X = X_train_genero
y = y_train_genero

print X.shape

tr = SelectKBest(chi2, k=8)
tr_f= SelectKBest(f_classif, k=8)

X_new = tr.fit_transform(X, y)

X_new_f = tr_f.fit_transform(X,y)

print X_new_f.shape
print X_new.shape

# ¿Qué características se han eliminado? Las marcadas con True en:
arr_chi2 = tr.get_support()
arr_anova =  tr_f.get_support()
print
print " chi2:" , X[X.columns[arr_chi2]].columns
print
print " anova: " , X[X.columns[arr_anova]].columns
```

```
(25298, 15)
(25298, 8)
(25298, 8)

 chi2: Index([u'email_server', u'idioma', u'os', u'pais', u'time_zone', u'b
rowser_family', u'device_family', u'id_hotspots'], dtype='object')

 anova:  Index([u'email_server', u'idioma', u'os', u'pais', u'time_zone',
u'browser_family', u'device_family', u'is_tounch_capable'], dtype='object')

/Users/Ana/anaconda/lib/python2.7/site-packages/sklearn/feature_selection/u
nivariate_selection.py:148: DeprecationWarning: Implicitly casting between
incompatible kinds. In a future numpy release, this will raise an error. Us
e casting="unsafe" if this is intentional.
  chisq -= f_exp
/Users/Ana/anaconda/lib/python2.7/site-packages/sklearn/feature_selection/u
nivariate_selection.py:150: DeprecationWarning: Implicitly casting between
incompatible kinds. In a future numpy release, this will raise an error. Us
e casting="unsafe" if this is intentional.
  chisq /= f_exp
```

Edad

In [24]:

```
X = X_train_edad
y = y_train_edad

print X.shape

tr = SelectKBest(chi2, k=8)
tr_f= SelectKBest(f_classif, k=8)

X_new = tr.fit_transform(X, y)

X_new_f = tr_f.fit_transform(X,y)

print X_new_f.shape
print X_new.shape

# ¿Qué características se han eliminado? Las marcadas con True en:
arr_chi2 = tr.get_support()
arr_anova =  tr_f.get_support()
print
print " Chi2:" , X[X.columns[arr_chi2]].columns
print
print " Anova: " , X[X.columns[arr_anova]].columns
```

```
(10895, 15)
(10895, 8)
(10895, 8)

 Chi2: Index([u'email_server', u'idioma', u'os', u'pais', u'browser_famil
y', u'device_family', u'is_tablet', u'id_hotspots'], dtype='object')

 Anova:  Index([u'email_server', u'idioma', u'os', u'browser_family', u'dev
ice_family', u'is_movile', u'is_tounch_capable', u'id_hotspots'], dtype='ob
ject')

/Users/Ana/anaconda/lib/python2.7/site-packages/sklearn/feature_selection/u
nivariate_selection.py:148: DeprecationWarning: Implicitly casting between
incompatible kinds. In a future numpy release, this will raise an error. Us
e casting="unsafe" if this is intentional.
  chisq -= f_exp
/Users/Ana/anaconda/lib/python2.7/site-packages/sklearn/feature_selection/u
nivariate_selection.py:150: DeprecationWarning: Implicitly casting between
incompatible kinds. In a future numpy release, this will raise an error. Us
e casting="unsafe" if this is intentional.
  chisq /= f_exp
```

10.- PCA Feature Reduction

Usamos todas las características

Primero tenemos que "binarizar las categorías", es decir, crear una columna por categoría y añadirla a dataFrame.

Vamsos a hacerlo con un bucle, para aquellas categorías que no son binarias

In [25]:

```
df.columns
```

Out[25]:

```
Index([u'ciudad', u'email_server', u'edad', u'genero', u'hora_visita', u'id
ioma', u'is_weekend', u'nombre_final', u'os', u'pais', u'rango_horario',
u'timezone', u'ua_browser_family', u'ua_device', u'ua_device_family', u'u
a_is_bot', u'ua_is_movile', u'ua_is_pc', u'ua_is_tablet', u'ua_is_tounch_ca
pable', u'ua_os_family', u'weekday', u'id_hotspots'], dtype='object')
```

In [26]:

```
#Creo diferentes arrays con los nombres de las columnas, los que son binari
os y los que no

cat_not_bin = ['email_server', 'hora_visita', 'idioma', 'os', 'pais', 'rang
o_horario',
               'timezone', 'ua_browser_family', 'ua_device_family', 'weekda
y', 'id_hotspots']

cat_bin=['is_weekend', 'ua_is_bot', 'ua_is_movile', 'ua_is_tounch_capable',
'ua_is_tablet']

#Y las columnas objetivo
cat_targets = ['edad', 'genero']
```

In [27]:

```
#Covierto las columnas boolean en 0 y 1

for col in df[cat_bin].columns:
    df.loc[:,(col)]  = [1 if x else 0 for x in df[col]]
```

Lo hago para género

In [28]:

```
#creo el nuevo dataframe y añado las categorías binarias

#Me creo unos dataframes con los datos que tengo que no son nulos para géne
ro y edad
df_genero_train = df[pd.notnull(df.genero)]
df_edad_train = df[pd.isnull(df.edad)]
df_genero_test = df[pd.notnull(df.genero)]
df_edad_test = df[pd.isnull(df.edad)]

# Covierto todo a tip "string" porque hay un bug en pandas, que no entiende
el tipo "category"
# a la hora de hacer un conctat

def convertType(data,tipo):
    for colum in data.columns:
        data.loc[:,(colum)] = data.loc[:,(colum)].astype(tipo)
    return data

df_genero_train = convertType(df_genero_train,'string')
df_edad_train = convertType(df_edad_train,'string')
df_genero_test = convertType(df_genero_test,'string')
df_edad_test = convertType(df_edad_test,'string')

print df_genero_train.dtypes
```

```
ciudad                  object
email_server            object
edad                    object
genero                  object
hora_visita             object
idioma                  object
is_weekend              object
nombre_final            object
os                      object
pais                    object
rango_horario           object
timezone                object
ua_browser_family       object
ua_device               object
ua_device_family        object
ua_is_bot               object
ua_is_movile            object
ua_is_pc                object
ua_is_tablet            object
ua_is_tounch_capable    object
ua_os_family            object
weekday                 object
id_hotspots             object
dtype: object
```

```
/Users/Ana/anaconda/lib/python2.7/site-packages/pandas/core/indexing.py:41
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 the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  self.obj[item] = s
```

In [29]:

```
# Función binariza que genera las columnas binarizadas
def binariza(dataf, cat_bin, cat_not_bin):
    df_binarized_genero = pd.DataFrame()
    df_binarized = dataf[cat_bin]
    #Y ahora añado las binarizadas

    for column in cat_not_bin:
        #Genero un arrat con las categorías que va a haber
        classes = df[column].unique().tolist()
        #Binarizo las columnas teniendo en cuenta las categorías

        column_bin = pp.label_binarize(dataf[column], classes)
        #y lo inserto en un dataframe dando nombre a las columnas
        df_bin = pd.DataFrame(column_bin,columns =
                                            ['is_'+ column + "_" + str(x).repl
ace(" ","_")
                                             for x in classes])

        #Como las variables binarizadas tienen un index distinto al de las
variables que ya
        #existían, al hacer el concat no se hace bien, por lo que ponemos e
l mismo índice
        #a las variables binarizadas que el que tenían las variables existe
ntes

        df_bin.index = df_binarized.index
        df_binarized = pd.concat((df_binarized,df_bin), axis=1)

    return df_binarized


df_genero_binarizado_train = binariza(df_genero_train, cat_bin, cat_not_bi
n)
df_edad_binarizado_train = binariza(df_edad_train, cat_bin, cat_not_bin)

df_genero_binarizado_test = binariza(df_genero_test, cat_bin, cat_not_bin)
df_edad_binarizado_test = binariza(df_edad_test, cat_bin, cat_not_bin)
```

In [30]:

```
#Vamos a ver las columnas que se han generado
print "PARA GÉNERO TRAIN: " , df_genero_binarizado_train.columns
print "PARA EDAD TRAIN: " ,df_edad_binarizado_train.columns

#y el tamaño de los dataframes
print "Tamaño Género Train: " ,len(df_genero_binarizado_train)
print "Tamaño Edad Train: " ,len(df_edad_binarizado_train)
```

PARA GÉNERO TRAIN: Index([u'is_weekend', u'ua_is_bot', u'ua_is_movile',
u'ua_is_tounch_capable', u'ua_is_tablet', u'is_email_server_0ramge.es', u'i
s_email_server_163.com', u'is_email_server_3dcrystalclear.com', u'is_emai
l_server_4a.ru', u'is_email_server_Arcor.de', u'is_email_server_GMAIL.COM',
u'is_email_server_Gmail.com', u'is_email_server_HOTMAIL.COM', u'is_email_se
rver_Hotmail.com', u'is_email_server_Hotmail.es', u'is_email_server_Puyuel
o.net', u'is_email_server_Yahoo.com.sg', u'is_email_server_a.com', u'is_ema
il_server_ab.com', u'is_email_server_abakus-media.de', u'is_email_server_ab
ogadosnavarro.com', u'is_email_server_abv.bg', u'is_email_server_adon.li',
u'is_email_server_aferrando.com', u'is_email_server_afiven.es', u'is_emai
l_server_ahora.es', u'is_email_server_aim.com', u'is_email_server_alejandro
resta.com', u'is_email_server_alexalcaide.com', u'is_email_server_alexepres
s.co.uj', u'is_email_server_alice.it', u'is_email_server_allenburt.plus.co
m', u'is_email_server_altiora.es', u'is_email_server_alu.uhu.es', u'is_emai
l_server_alumni.uv.es', u'is_email_server_alumnos.uchceu.es', u'is_email_se
rver_andaluciajunta.es', u'is_email_server_andamiosmadrid.es', u'is_email_s
erver_aol.co.uk', u'is_email_server_aol.com', u'is_email_server_aol.de',
u'is_email_server_apt-plandevelop.co.uk', u'is_email_server_arcor.de', u'i
s_email_server_arraiz.es', u'is_email_server_arrakis.es', u'is_email_serve
r_ascensiresfit.es', u'is_email_server_ascensoresfit.com', u'is_email_serve
r_asf.com', u'is_email_server_atlas.cz', u'is_email_server_avory.es', u'i
s_email_server_awe.be', u'is_email_server_axu-renting.es', u'is_email_serve
r_baoproyectos.com', u'is_email_server_bardon.com', u'is_email_server_bcg.c
om', u'is_email_server_bellsouth.net', u'is_email_server_berklee.edu', u'i
s_email_server_bk.ru', u'is_email_server_blagushin.ru', u'is_email_server_b
lu.it', u'is_email_server_bluewin.ch', u'is_email_server_blueyonder.co.uk',
u'is_email_server_bmcinnovation.com', u'is_email_server_bodegatrasiegos.e
s', u'is_email_server_boston.com.ar', u'is_email_server_bradleyphysio.co.u
k', u'is_email_server_brinterney.com', u'is_email_server_brotherswing.com',
u'is_email_server_brotons.net', u'is_email_server_btconnect.com', u'is_emai
l_server_btinterbet.com', u'is_email_server_btinternet.c', u'is_email_serve
r_btinternet.com', u'is_email_server_btintwrnet.com', u'is_email_server_bto
penworld.com', u'is_email_server_bundestag.de', u'is_email_server_caliburn-
software.com', u'is_email_server_carrier.se', u'is_email_server_cartujaspor
t.com', u'is_email_server_casema.nl', u'is_email_server_castellanagolf.co
m', u'is_email_server_castvalencia.es', u'is_email_server_ccg.nu', u'is_ema
il_server_centrum.cz', u'is_email_server_chronodrive.com', u'is_email_serve
r_cityweb.de', u'is_email_server_claregalwayhotel.ie', u'is_email_server_cl
ub-internet.fr', u'is_email_server_cngservices.co.uk', u'is_email_server_co
ev.com', u'is_email_server_coitihuelva.com', u'is_email_server_colesan.ed
u.co', u'is_email_server_colorbar.es', u'is_email_server_comcast.net', u'i
s_email_server_compuserve.com', u'is_email_server_configur8or.com', u'is_em
ail_server_covidien.com', u'is_email_server_cox.ch', u'is_email_server_crit
icker.com', u'is_email_server_cs.com', ...], dtype='object')
PARA EDAD TRAIN: Index([u'is_weekend', u'ua_is_bot', u'ua_is_movile', u'u
a_is_tounch_capable', u'ua_is_tablet', u'is_email_server_0ramge.es', u'is_e
mail_server_163.com', u'is_email_server_3dcrystalclear.com', u'is_email_ser
ver_4a.ru', u'is_email_server_Arcor.de', u'is_email_server_GMAIL.COM', u'i
s_email_server_Gmail.com', u'is_email_server_HOTMAIL.COM', u'is_email_serve
r_Hotmail.com', u'is_email_server_Hotmail.es', u'is_email_server_Puyuelo.ne
t', u'is_email_server_Yahoo.com.sg', u'is_email_server_a.com', u'is_email_s
erver_ab.com', u'is_email_server_abakus-media.de', u'is_email_server_abogad
osnavarro.com', u'is_email_server_abv.bg', u'is_email_server_adon.li', u'i
s_email_server_aferrando.com', u'is_email_server_afiven.es', u'is_email_ser
ver_ahora.es', u'is_email_server_aim.com', u'is_email_server_alejandrorest
a.com', u'is_email_server_alexalcaide.com', u'is_email_server_alexepress.c
o.uj', u'is_email_server_alice.it', u'is_email_server_allenburt.plus.com',

```
u'is_email_server_altiora.es', u'is_email_server_alu.uhu.es', u'is_email_se
rver_alumni.uv.es', u'is_email_server_alumnos.uchceu.es', u'is_email_serve
r_andaluciajunta.es', u'is_email_server_andamiosmadrid.es', u'is_email_serv
er_aol.co.uk', u'is_email_server_aol.com', u'is_email_server_aol.de', u'i
s_email_server_apt-plandevelop.co.uk', u'is_email_server_arcor.de', u'is_em
ail_server_arraiz.es', u'is_email_server_arrakis.es', u'is_email_server_asc
ensiresfit.es', u'is_email_server_ascensoresfit.com', u'is_email_server_as
f.com', u'is_email_server_atlas.cz', u'is_email_server_avory.es', u'is_emai
l_server_awe.be', u'is_email_server_axu-renting.es', u'is_email_server_baop
royectos.com', u'is_email_server_bardon.com', u'is_email_server_bcg.com',
u'is_email_server_bellsouth.net', u'is_email_server_berklee.edu', u'is_emai
l_server_bk.ru', u'is_email_server_blagushin.ru', u'is_email_server_blu.i
t', u'is_email_server_bluewin.ch', u'is_email_server_blueyonder.co.uk', u'i
s_email_server_bmcinnovation.com', u'is_email_server_bodegatrasiegos.es',
u'is_email_server_boston.com.ar', u'is_email_server_bradleyphysio.co.uk',
u'is_email_server_brinterney.com', u'is_email_server_brotherswing.com', u'i
s_email_server_brotons.net', u'is_email_server_btconnect.com', u'is_email_s
erver_btinterbet.com', u'is_email_server_btinternet.c', u'is_email_server_b
tinternet.com', u'is_email_server_btintwrnet.com', u'is_email_server_btopen
world.com', u'is_email_server_bundestag.de', u'is_email_server_caliburn-sof
tware.com', u'is_email_server_carrier.se', u'is_email_server_cartujasport.c
om', u'is_email_server_casema.nl', u'is_email_server_castellanagolf.com',
u'is_email_server_castvalencia.es', u'is_email_server_ccg.nu', u'is_email_s
erver_centrum.cz', u'is_email_server_chronodrive.com', u'is_email_server_ci
tyweb.de', u'is_email_server_claregalwayhotel.ie', u'is_email_server_club-i
nternet.fr', u'is_email_server_cngservices.co.uk', u'is_email_server_coev.c
om', u'is_email_server_coitihuelva.com', u'is_email_server_colesan.edu.co',
u'is_email_server_colorbar.es', u'is_email_server_comcast.net', u'is_emai
l_server_compuserve.com', u'is_email_server_configur8or.com', u'is_email_se
rver_covidien.com', u'is_email_server_cox.ch', u'is_email_server_criticke
r.com', u'is_email_server_cs.com', ...], dtype='object')
Tamaño Género Train:  25298
Tamaño Edad Train:  15183
```

y comienzo con la reducción de variables

In [31]:

```
from sklearn.decomposition import PCA
```

Hay que convertir todo a float64 para poder hacer la reducción de variables

In [32]:

```
print df_genero_binarizado_test.head()

df_genero_binarizado_train = convertType(df_genero_binarizado_train,'float6
4')
df_edad_binarizado_train = convertType(df_edad_binarizado_train,'float64')

df_genero_binarizado_test = convertType(df_genero_binarizado_test,'float6
4')
df_edad_binarizado_test = convertType(df_edad_binarizado_test,'float64')
```

```
   is_weekend ua_is_bot ua_is_movile ua_is_tounch_capable ua_is_tablet  \
9           1         0            1                    1            0
10          1         0            1                    1            0
11          1         0            1                    1            0
12          1         0            1                    1            0
13          1         0            1                    1            0

    is_email_server_0ramge.es  is_email_server_163.com  \
9                           0                        0
10                          0                        0
11                          0                        0
12                          0                        0
13                          0                        0

    is_email_server_3dcrystalclear.com  is_email_server_4a.ru  \
9                                    0                      0
10                                   0                      0
11                                   0                      0
12                                   0                      0
13                                   0                      0

    is_email_server_Arcor.de         ...           is_id_hotspots_27.0  \
9                          0         ...                             0
10                         0         ...                             0
11                         0         ...                             0
12                         0         ...                             0
13                         0         ...                             0

    is_id_hotspots_28.0  is_id_hotspots_29.0  is_id_hotspots_30.0  \
9                     0                    0                    0
10                    0                    0                    0
11                    0                    0                    0
12                    0                    0                    0
13                    0                    0                    0

    is_id_hotspots_31.0  is_id_hotspots_32.0  is_id_hotspots_33.0  \
9                     0                    0                    0
10                    0                    0                    0
11                    0                    0                    0
12                    0                    0                    0
13                    0                    0                    0

    is_id_hotspots_34.0  is_id_hotspots_35.0  is_id_hotspots_36.0
9                     0                    0                    0
10                    0                    0                    0
11                    0                    0                    0
12                    0                    0                    0
13                    0                    0                    0

[5 rows x 1274 columns]
```

Género

In [33]:

```
n_features = df_genero_binarizado_train.columns.size
print "Total number of features for género: %d" %n_features
```

```
Total number of features for género: 1274
```

In [34]:

```
pca = PCA(n_components=n_features, whiten=False)
pca.fit(df_genero_binarizado_train)
```

Out[34]:

```
PCA(copy=True, n_components=1274, whiten=False)
```

In [35]:

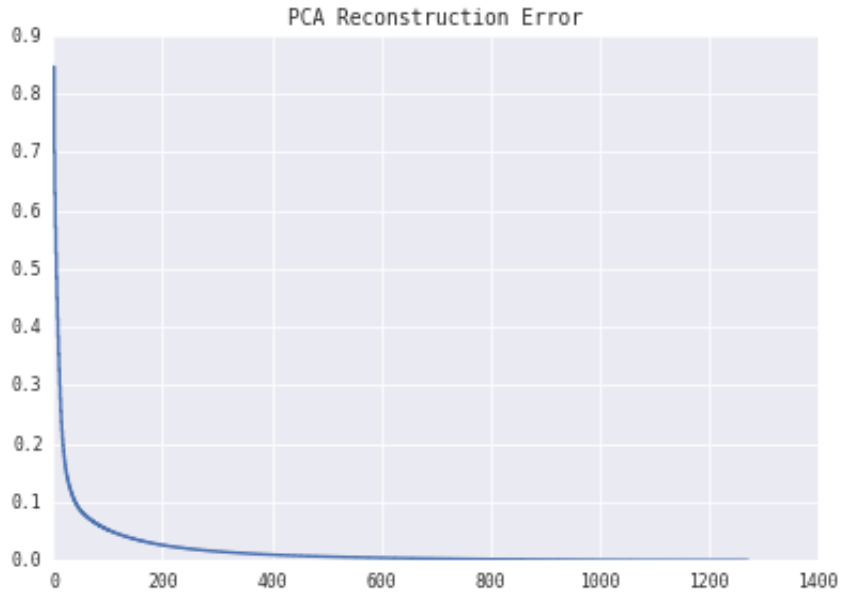
```
pca.explained_variance_ratio_[0:].cumsum()
```

Out[35]:

```
array([ 0.15445939,  0.30052129,  0.37111401, ...,  1.        ,
        1.        ,  1.        ])
```

In [36]:

```
plt.plot(1 - pca.explained_variance_ratio_.cumsum(), drawstyle = 'steps-pos
t')
plt.title('PCA Reconstruction Error');
```



Vemos el número de variables que necesitamos para explicar el 90% de la varianza

In [37]:

```
n_factors = sum(1-pca.explained_variance_ratio_[0:].cumsum() > 0.10)
print "Number of factors with 10% of reconstraction Error: ", n_factors
```

```
Number of factors with 10% of reconstraction Error:  40
```

Entonces entrenamos el algoritmo con el número de varibles que se nos indica:

In [38]:

```
n_factors
```

Out[38]:

```
40
```

In [39]:

```
pca = PCA(n_components=n_factors)
pca.fit(df_edad_binarizado_train)
```

Out[39]:

```
PCA(copy=True, n_components=40, whiten=False)
```

In [40]:

```
print "Explained Variance Ratio"
print sum(pca.explained_variance_ratio_)
```

```
Explained Variance Ratio
0.908183192795
```

In [41]:

```
trainDS_pca = pca.transform(df_edad_binarizado_train)
```