Clase Feature Selection usando f-score y mutual information

August 5, 2017

In [21]:
```
# Feature Extraction with Univariate Statistical Tests (Chi-squared for classificatio
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest,f_regression,mutual_info_regression
from sklearn.model_selection import train_test_split,cross_val_score
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from sklearn.preprocessing import scale
#Para graficar
import matplotlib.pyplot as plt
```

In [17]:
```
df=pd.read_csv('funciones.csv')
del df['Unnamed: 0']
```

In [18]:
```
#forma rapida de contar los valores nuloes o "missin values" en las variables como % 
#pleta de total minutes las variables de nunca superan el 1.1% de valores NA en el re
#(un máximo de 78 datos por variable). Guardamos la data completa en nuestro backup d
dfnum=df.dropna(axis=0,how='any',thresh=None,subset=['total_minutes'])
round(100*(df.isnull().sum(axis=0)/df.shape[0]),1)
```

Out[18]:
```
total_minutes            0.0
min_dif                  0.0
found_rate_pickers       1.1
picking_speed_pickers    0.0
accepted_rate_pickers    0.1
rating_pickers           1.1
found_rate_drivers       1.2
picking_speed_drivers    0.0
accepted_rate_drivers    0.1
rating_drivers           1.1
quantity_Kg             38.0
found_rate_Kg           38.0
quantity_UN              0.7
found_rate_UN            0.7
dtype: float64
```

In [19]:
```
dfnum=dfnum.loc[dfnum[((dfnum['quantity_UN'].isnull()) & (dfnum['quantity_Kg'].isnull
#Ahora imputamos los valores NA restantes como como 0 en las variables de quantity y 
dfnum[['quantity_UN','found_rate_UN','quantity_Kg','found_rate_Kg']]=dfnum[['quantity
round(100*(dfnum.isnull().sum(axis=0)/dfnum.shape[0]),1)
```

Out[19]:
```
total_minutes            0.0
min_dif                  0.0
found_rate_pickers       1.1
picking_speed_pickers    0.0
accepted_rate_pickers    0.1
rating_pickers           1.1
found_rate_drivers       1.2
picking_speed_drivers    0.0
accepted_rate_drivers    0.1
rating_drivers           1.1
quantity_Kg              0.0
found_rate_Kg            0.0
quantity_UN              0.0
found_rate_UN            0.0
dtype: float64
```

In [20]:
```
#Concretamos el punto 3
X_scaled = scale(np.array(dfnum['quantity_UN']))+scale(np.array(dfnum['quantity_Kg']))
dfnum['q_total']=X_scaled
X_scaled = (np.array(dfnum['found_rate_Kg'])+np.array(dfnum['found_rate_UN']))/2
dfnum['found_rate_avg']=X_scaled
#Vemos una correlación relevtante entre q_total con total_minutes(0.62), así como tam
#débil pero de todas formas relevantes en las variables de quantity entre si.
cm=dfnum.corr()
print(cm['q_total'])
sbs.heatmap(cm,square=True)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-20-1dae7a429111> in <module>()
      1 #Concretamos el punto 3
----> 2 X_scaled = scale(np.array(dfnum['quantity_UN']))+scale(np.array(dfnum['quantity_Kg
      3 dfnum['q_total']=X_scaled
      4 X_scaled = (np.array(dfnum['found_rate_Kg'])+np.array(dfnum['found_rate_UN']))/2
      5 dfnum['found_rate_avg']=X_scaled


NameError: name 'scale' is not defined
```

In [5]:
```
#Ahora comenzaremos a usar la librería sklearn para hacer featured selection entre las
#la variable objetivo, que en este caso será el tiempo total de entrega.Necesitamos  t
#arreglos y separar el dataset con los potenciales features y la variable target
#Primero usaremos la selección de variables continueas y postivias usando regresión e
#http://powerhousedm.blogspot.cl/2008/10/correlacin-e-informacin-mutua.html
#https://es.wikipedia.org/wiki/Información_mutua
```

In [25]:
```
#Usando dataset sin NA's para generar un ranking basado en
X=dfnum.dropna(axis=0,how='any').loc[:,dfnum.columns.values[2:len(dfnum.columns.values
Y=dfnum.dropna(axis=0,how='any').loc[:,['total_minutes']].values.ravel().astype(float
#Divide into training and test-set
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=
#Ahora usamos el la función bestSelect que incorpora el parámetro center=True para qu
#Luego el selectKbest los selecciona por F-score
test = SelectKBest(score_func=f_regression,k=4)
fit = test.fit(X=X_train,y=y_train)
# summarize scores
np.set_printoptions(precision=3)
print(fit.scores_)
features = fit.transform(X_train)
# summarize selected features
print(features[0:4,:])
#como podemos ver en los resultados, arroja los primeros 4 valores de quantity para U
#picking_speed_drivers, pues nosotros solicitamos que nos entregara los mejores 4.

#Mostramos los resultados de forma ordenada
arr1=dfnum.columns.values[2:len(dfnum.columns.values)]
arr2=fit.scores_
arr3=arr2/np.max(arr2)

#compramos con mutual information
mi = mutual_info_regression(X_train, y_train)
arr4=mi
arr5= arr4/np.max(arr4)

#Dibujamos
raw_data={'features':arr1,'f-score':arr2,'f-weight':arr3,'m-score':arr4,'m-weight':arr
features=pd.DataFrame(raw_data,columns=['features','f-score','f-weight','m-score','m-w
features.sort_values(by='m-weight',ascending=False)
```

```
[  2.941e+00   1.393e+01   2.953e-01   1.629e+00   1.423e+00   1.539e+01
   2.644e-01   1.355e+00   1.547e+03   8.039e+01   2.787e+03   1.043e+01]
[[  2.      0.      0.      1.   ]
 [  1.63    0.      0.      5.   ]
 [  1.57   14.35    1.038  75.   ]
 [  1.58    6.1     1.195  30.   ]]
```

Out[25]:

	features	f-score	f-weight	m-score	m-weight
10	quantity_UN	2786.734047	1.000000	0.244899	1.000000
8	quantity_Kg	1546.826786	0.555068	0.143409	0.585586
11	found_rate_UN	10.431806	0.003743	0.088607	0.361811
9	found_rate_Kg	80.385527	0.028846	0.071710	0.292816
1	picking_speed_pickers	13.933363	0.005000	0.026985	0.110191
5	picking_speed_drivers	15.387470	0.005522	0.022541	0.092040

```
0      found_rate_pickers     2.941236  0.001055  0.014874  0.060735
4      found_rate_drivers     1.422664  0.000511  0.011651  0.047574
3          rating_pickers     1.628534  0.000584  0.009520  0.038874
7          rating_drivers     1.355397  0.000486  0.001744  0.007122
2   accepted_rate_pickers     0.295330  0.000106  0.000000  0.000000
6   accepted_rate_drivers     0.264436  0.000095  0.000000  0.000000
```

In [26]:
```
df.loc[:,['found_rate_pickers','picking_speed_pickers','accepted_rate_pickers','rating
      'picking_speed_drivers', 'accepted_rate_drivers','rating_drivers']].head()
```

Out[26]:

	found_rate_pickers	picking_speed_pickers	accepted_rate_pickers	\
0	0.8564	1.56	1.00	
1	0.8516	1.14	1.00	
2	0.8337	2.03	1.00	
3	0.8571	2.06	0.92	
4	0.8625	1.89	1.00	

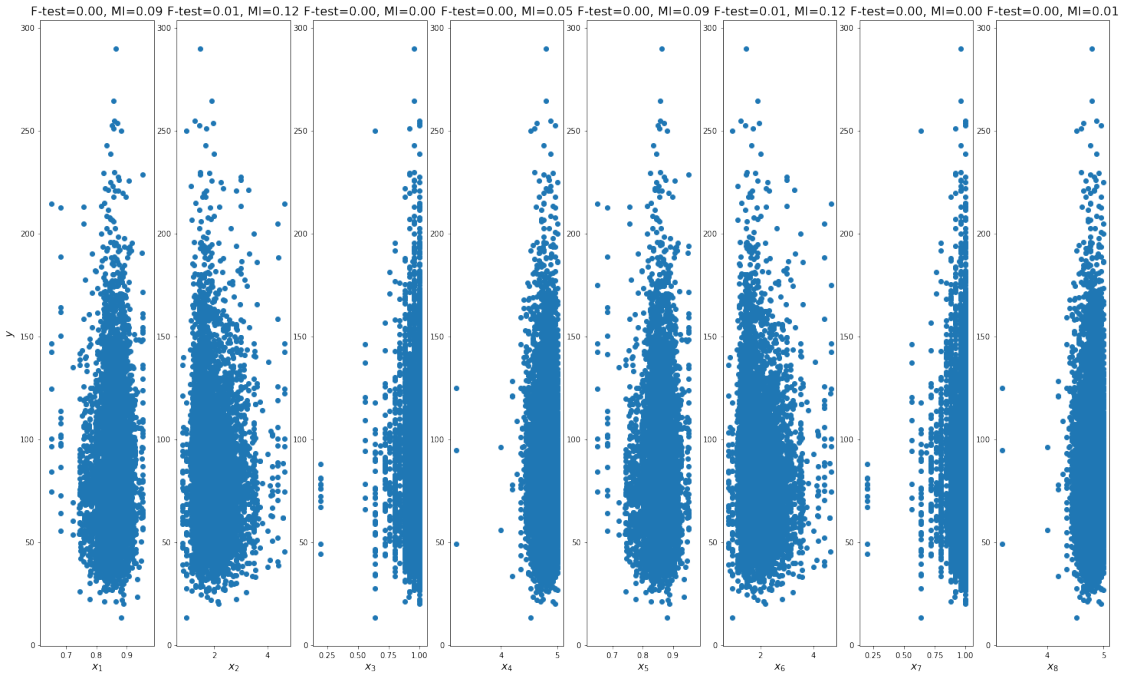
	rating_pickers	found_rate_drivers	picking_speed_drivers	\
0	4.68	0.8564	1.56	
1	4.76	0.8516	1.14	
2	4.96	0.8337	2.03	
3	4.92	0.8571	2.06	
4	4.92	0.8625	1.89	

	accepted_rate_drivers	rating_drivers
0	1.00	4.68
1	1.00	4.76
2	1.00	4.96
3	0.92	4.92
4	1.00	4.92

In [27]:
```
f_test, _ = f_regression(X=X,y=Y)
f_test /= np.max(f_test)

mi = mutual_info_regression(X=X,y=Y)
mi /= np.max(mi)

plt.figure(figsize=(25, 15))
for i in range(8):
    plt.subplot(1, 8, i + 1)
    plt.scatter(X[:, i], Y)
    plt.xlabel("$x_{}$".format(i + 1), fontsize=14)
    if i == 0:
        plt.ylabel("$y$", fontsize=14)
    plt.title("F-test={:.2f}, MI={:.2f}".format(f_test[i], mi[i]),
              fontsize=16)
plt.show()
```



In [ ]:
```
#Seleccionamos con certeza las variables de picking speed y found rate tanto de picker
#relevantes de los shopper en el análisis del tiempo total de demora.
```