2017_session6

May 17, 2019

1 Session 26/6/2017 - machine learning

Découverte des trois problèmes de machine learning exposé dans l'article Machine Learning - session 6.

```
In [1]: from jyquickhelper import add_notebook_menu
        add_notebook_menu()
```

```
Out[1]: <IPython.core.display.HTML object>
```

1.1 Problème 1 : comparaison random forest, linéaire

C'est un problème de régression. On cherche à comparer une random forest avec un modèle linéaire.

- Comparaison des tests de coefficients pour un modèle linéaire OLS et des features importance
- Résultat au niveau d'une observation treeinterpreter
- Données : Housing, Forest Fire

1.1.1 Données

```
In [2]: import pandas
        df = pandas.read_csv("data/housing.data", delim_whitespace=True, header=None)
        df.head()
```

```
Out[2]:         0     1     2   3      4      5     6       7   8      9     10  \
        0  0.00632  18.0  2.31   0  0.538  6.575  65.2  4.0900   1  296.0  15.3
        1  0.02731   0.0  7.07   0  0.469  6.421  78.9  4.9671   2  242.0  17.8
        2  0.02729   0.0  7.07   0  0.469  7.185  61.1  4.9671   2  242.0  17.8
        3  0.03237   0.0  2.18   0  0.458  6.998  45.8  6.0622   3  222.0  18.7
        4  0.06905   0.0  2.18   0  0.458  7.147  54.2  6.0622   3  222.0  18.7

               11    12    13
        0  396.90  4.98  24.0
        1  396.90  9.14  21.6
        2  392.83  4.03  34.7
        3  394.63  2.94  33.4
        4  396.90  5.33  36.2
```

```
In [3]: cols = "CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX PTRATIO B LSTAT MEDV".split()
        df.columns = cols
        df.head()
```

```
Out[3]:       CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
        0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0
        1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0
```

```
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0

   PTRATIO       B  LSTAT  MEDV
0     15.3  396.90   4.98  24.0
1     17.8  396.90   9.14  21.6
2     17.8  392.83   4.03  34.7
3     18.7  394.63   2.94  33.4
4     18.7  396.90   5.33  36.2
```

```
In [4]: X = df.drop("MEDV", axis=1)
        y = df["MEDV"]
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

1.1.2 Random Forest

```
In [5]: from sklearn.ensemble import RandomForestRegressor
        clr = RandomForestRegressor()
        clr.fit(X, y)
```

```
c:\python370_x64\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: DeprecationWarning: numpy.co
  from numpy.core.umath_tests import inner1d
```

```
Out[5]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                    max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                    oob_score=False, random_state=None, verbose=0, warm_start=False)
```

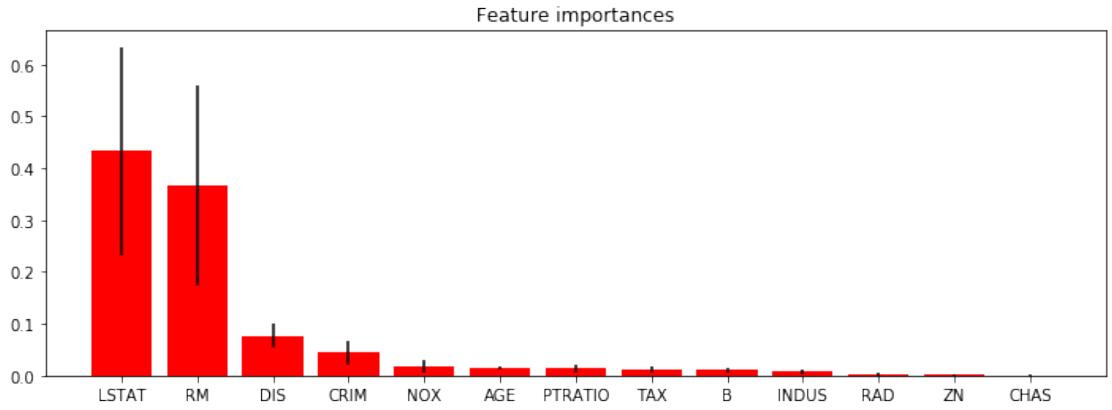
```
In [6]: importances = clr.feature_importances_
        importances
```

```
Out[6]: array([0.0443893 , 0.00110388, 0.00710562, 0.00068142, 0.01735106,
               0.36666861, 0.01430106, 0.07699777, 0.00330815, 0.01167688,
               0.01337202, 0.01036104, 0.4326832 ])
```

On s'inspire de l'exemple Feature importances with forests of trees.

```
In [7]: %matplotlib inline
```

```
In [8]: import matplotlib.pyplot as plt
        import numpy as np
        plt.figure(figsize=(12,4))
        indices = np.argsort(importances)[::-1]
        std = np.std([tree.feature_importances_ for tree in clr.estimators_],
                     axis=0)
        plt.title("Feature importances")
        plt.bar(range(X.shape[1]), importances[indices],
                color="r", yerr=std[indices], align="center")
        xlabels = list(df.columns[:-1])
        xlabels = [xlabels[i] for i in indices]
        plt.xticks(range(X.shape[1]), xlabels)
        plt.xlim([-1, X.shape[1]])
        plt.show()
```




```
In [9]: from sklearn.metrics import r2_score
        r2_score(y_train, clr.predict(X_train))

Out[9]: 0.971141486563686

In [10]: r2_score(y_test, clr.predict(X_test))

Out[10]: 0.9858344257401853
```

1.1.3 Modèle linéaire

```
In [11]: import statsmodels.api as sm

In [12]: model = sm.OLS(y_train, X_train)

In [13]: results = model.fit()

In [14]: results.params

Out[14]: CRIM      -0.076817
         ZN         0.057742
         INDUS     -0.038642
         CHAS       3.675863
         NOX       -2.731249
         RM         5.741715
         AGE       -0.012431
         DIS       -1.071296
         RAD        0.151342
         TAX       -0.008279
         PTRATIO   -0.333420
         B          0.015925
         LSTAT     -0.398157
         dtype: float64

In [15]: results.summary()

Out[15]: <class 'statsmodels.iolib.summary.Summary'>
         """
                                    OLS Regression Results
```

```
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.958
Model:                            OLS   Adj. R-squared:                  0.957
Method:                 Least Squares   F-statistic:                     576.6
Date:                Mon, 03 Sep 2018   Prob (F-statistic):          5.05e-216
Time:                        16:59:26   Log-Likelihood:                -1028.9
No. Observations:                 339   AIC:                             2084.
Df Residuals:                     326   BIC:                             2134.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
CRIM          -0.0768      0.040     -1.901      0.058      -0.156       0.003
ZN             0.0577      0.019      3.088      0.002       0.021       0.095
INDUS         -0.0386      0.085     -0.457      0.648      -0.205       0.128
CHAS           3.6759      1.063      3.459      0.001       1.585       5.767
NOX           -2.7312      4.173     -0.655      0.513     -10.940       5.478
RM             5.7417      0.386     14.893      0.000       4.983       6.500
AGE           -0.0124      0.017     -0.727      0.468      -0.046       0.021
DIS           -1.0713      0.250     -4.285      0.000      -1.563      -0.579
RAD            0.1513      0.083      1.820      0.070      -0.012       0.315
TAX           -0.0083      0.005     -1.652      0.099      -0.018       0.002
PTRATIO       -0.3334      0.140     -2.389      0.017      -0.608      -0.059
B              0.0159      0.004      4.477      0.000       0.009       0.023
LSTAT         -0.3982      0.064     -6.228      0.000      -0.524      -0.272
==============================================================================
Omnibus:                      128.638   Durbin-Watson:                   1.899
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              849.018
Skew:                           1.421   Prob(JB):                    4.35e-185
Kurtosis:                      10.213   Cond. No.                     8.43e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.43e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

In [16]:
```
model = sm.OLS(y,X.drop("LSTAT", axis=1))
results = model.fit()
results.summary()
```

Out[16]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.954
Model:                            OLS   Adj. R-squared:                  0.953
Method:                 Least Squares   F-statistic:                     846.6
Date:                Mon, 03 Sep 2018   Prob (F-statistic):          2.38e-320
Time:                        16:59:26   Log-Likelihood:                -1556.1
No. Observations:                 506   AIC:                             3136.
Df Residuals:                     494   BIC:                             3187.
Df Model:                          12
```

```
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
CRIM          -0.1439      0.036     -3.990      0.000      -0.215      -0.073
ZN             0.0413      0.015      2.696      0.007       0.011       0.071
INDUS         -0.0370      0.068     -0.540      0.589      -0.172       0.098
CHAS           3.2525      0.961      3.384      0.001       1.364       5.141
NOX          -10.8653      3.422     -3.175      0.002     -17.590      -4.141
RM             7.1436      0.289     24.734      0.000       6.576       7.711
AGE           -0.0449      0.014     -3.235      0.001      -0.072      -0.018
DIS           -1.2292      0.206     -5.980      0.000      -1.633      -0.825
RAD            0.2008      0.071      2.829      0.005       0.061       0.340
TAX           -0.0100      0.004     -2.391      0.017      -0.018      -0.002
PTRATIO       -0.6575      0.112     -5.881      0.000      -0.877      -0.438
B              0.0165      0.003      5.779      0.000       0.011       0.022
==============================================================================
Omnibus:                      277.013   Durbin-Watson:                   0.927
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3084.310
Skew:                           2.148   Prob(JB):                         0.00
Kurtosis:                      14.307   Cond. No.                     8.13e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.13e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

1.1.4 TPOT

TPOT est un module d'apprentissage automatique.

```
In [17]: from tpot import TPOTRegressor
         tpot = TPOTRegressor(generations=2, population_size=50, verbosity=2)
         tpot.fit(X_train, y_train)
         print(tpot.score(X_test, y_test))
         tpot.export('tpot_boston_pipeline.py')
```

```
Generation 1 - Current best internal CV score: -14.668309970410537
```

```
Generation 2 - Current best internal CV score: -12.57757387796134
```

```
Best pipeline: RandomForestRegressor(PolynomialFeatures(input_matrix, degree=2, include_bias=False, int
-8.106061359660954
```

```
Out[17]: True
```

Le module optimise les hyperparamètres, parfois un peu trop à en juger la mauvaise performance obtenue sur la base de test.

```
In [18]: r2_score(y_train, tpot.predict(X_train))

Out[18]: 0.9940508850756388

In [19]: r2_score(y_test, tpot.predict(X_test))

Out[19]: 0.8980064603926491
```

1.1.5 Feature importance pour une observations

On reprend la première random forest et on utilise le module treeinterpreter.

```
In [20]: clr = RandomForestRegressor()
         clr.fit(X, y)

Out[20]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                    max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                    oob_score=False, random_state=None, verbose=0, warm_start=False)

In [21]: from treeinterpreter import treeinterpreter as ti
         prediction, bias, contributions = ti.predict(clr, X_test)

In [22]: for i in range(min(2, X_train.shape[0])):
             print("Instance", i)
             print("Bias (trainset mean)", bias[i])
             print("Feature contributions:")
             for c, feature in sorted(zip(contributions[i], df.columns),
                                      key=lambda x: -abs(x[0])):
                 print(feature, round(c, 2))
             print( "-"*20)

Instance 0
Bias (trainset mean) 22.509664031620556
Feature contributions:
LSTAT -8.26
RM -1.64
CRIM -1.54
NOX -1.25
B -0.52
TAX -0.41
DIS -0.23
AGE 0.07
ZN 0.0
INDUS 0.0
CHAS 0.0
RAD 0.0
PTRATIO 0.0
--------------------
```

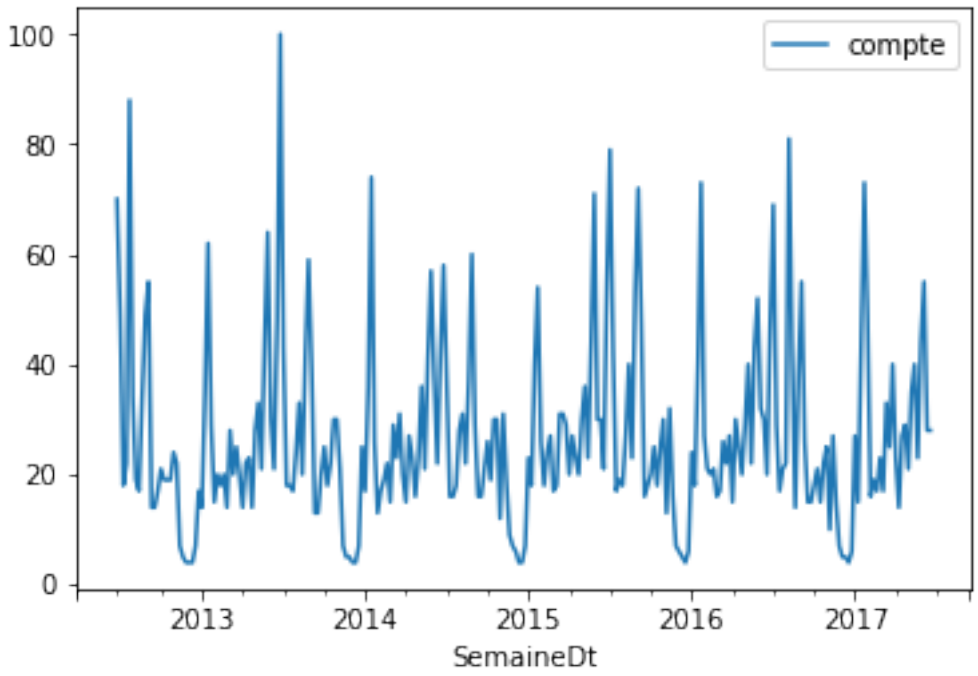
```
Instance 1
Bias (trainset mean) 22.509664031620556
Feature contributions:
LSTAT 5.21
RM -3.79
DIS -0.27
AGE -0.23
B -0.2
INDUS 0.12
PTRATIO 0.11
NOX -0.09
ZN -0.08
CRIM 0.03
RAD 0.02
CHAS -0.01
TAX 0.0
--------------------
```

1.2 Problème 2 : série temporelle

On prend une série sur Google Trends, dans notre cas, c'est la requête *tennis live*. On compare une approche linéaire et une approche non linéaire.

1.2.1 Approche linéaire

```
In [23]: import pandas
         df = pandas.read_csv("data/multiTimeline.csv", skiprows=1)

In [24]: df.columns= ["Semaine", "compte"]

In [25]: df["SemaineDt"] = pandas.to_datetime(df.Semaine)

In [26]: df=df.set_index("SemaineDt")

In [27]: df["compte"] = df["compte"].astype(float)

In [28]: df.head()

Out[28]:                Semaine  compte
         SemaineDt
         2012-07-01  2012-07-01    70.0
         2012-07-08  2012-07-08    49.0
         2012-07-15  2012-07-15    18.0
         2012-07-22  2012-07-22    22.0
         2012-07-29  2012-07-29    88.0

In [29]: %matplotlib inline

In [30]: df.plot()

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x284e690b710>
```




```
In [31]: from statsmodels.tsa.arima_model import ARIMA
         arma_mod = ARIMA(df["compte"].as_matrix(), order=(6 ,1, 1))
         res = arma_mod.fit()
         res.params

c:\python370_x64\lib\site-packages\ipykernel_launcher.py:2: FutureWarning: Method .as_matrix will be re

Out[31]: array([ 0.0041858 ,  0.59035768, -0.3254069 ,  0.2328683 , -0.03300863,
                 0.06434328, -0.07204005, -0.99999992])

In [32]: res.summary()

Out[32]: <class 'statsmodels.iolib.summary.Summary'>
         """
                                      ARIMA Model Results
         ==============================================================================
         Dep. Variable:                    D.y   No. Observations:                  259
         Model:                 ARIMA(6, 1, 1)   Log Likelihood               -1055.581
         Method:                       css-mle   S.D. of innovations             14.116
         Date:                Mon, 03 Sep 2018   AIC                           2129.161
         Time:                        17:00:22   BIC                           2161.173
         Sample:                             1   HQIC                          2142.032

         ==============================================================================
                         coef    std err          z      P>|z|      [0.025      0.975]
         ------------------------------------------------------------------------------
         const          0.0042      0.021      0.196      0.845      -0.038       0.046
```

```
ar.L1.D.y      0.5904      0.063      9.431      0.000       0.468       0.713
ar.L2.D.y     -0.3254      0.072     -4.507      0.000      -0.467      -0.184
ar.L3.D.y      0.2329      0.075      3.097      0.002       0.085       0.380
ar.L4.D.y     -0.0330      0.076     -0.433      0.665      -0.182       0.116
ar.L5.D.y      0.0643      0.076      0.842      0.400      -0.085       0.214
ar.L6.D.y     -0.0720      0.066     -1.096      0.274      -0.201       0.057
ma.L1.D.y     -1.0000      0.010    -96.075      0.000      -1.020      -0.980
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -1.2011           -1.2144j            1.7080           -0.3741
AR.2           -1.2011           +1.2144j            1.7080            0.3741
AR.3            0.1840           -1.4018j            1.4138           -0.2292
AR.4            0.1840           +1.4018j            1.4138            0.2292
AR.5            1.4636           -0.4882j            1.5429           -0.0512
AR.6            1.4636           +0.4882j            1.5429            0.0512
MA.1            1.0000           +0.0000j            1.0000            0.0000
-----------------------------------------------------------------------------
"""
```

1.2.2 Méthode non linéaire

On construire la matrice des séries décalées. Cette méthode permet de sortir du cadre linéaire et d'ajouter d'autres variables.

```
In [33]: from statsmodels.tsa.tsatools import lagmat
         lag = 8
         X = lagmat(df["compte"], lag)
         lagged = df.copy()
         for c in range(1,lag+1):
             lagged["lag%d" % c] = X[:, c-1]
         pandas.concat([lagged.head(), lagged.tail()])
```

Out[33]:

	Semaine	compte	lag1	lag2	lag3	lag4	lag5	lag6	lag7	lag8
SemaineDt										
2012-07-01	2012-07-01	70.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2012-07-08	2012-07-08	49.0	70.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2012-07-15	2012-07-15	18.0	49.0	70.0	0.0	0.0	0.0	0.0	0.0	0.0
2012-07-22	2012-07-22	22.0	18.0	49.0	70.0	0.0	0.0	0.0	0.0	0.0
2012-07-29	2012-07-29	88.0	22.0	18.0	49.0	70.0	0.0	0.0	0.0	0.0
2017-05-21	2017-05-21	23.0	40.0	35.0	21.0	29.0	27.0	14.0	23.0	40.0
2017-05-28	2017-05-28	44.0	23.0	40.0	35.0	21.0	29.0	27.0	14.0	23.0
2017-06-04	2017-06-04	55.0	44.0	23.0	40.0	35.0	21.0	29.0	27.0	14.0
2017-06-11	2017-06-11	28.0	55.0	44.0	23.0	40.0	35.0	21.0	29.0	27.0
2017-06-18	2017-06-18	28.0	28.0	55.0	44.0	23.0	40.0	35.0	21.0	29.0

```
In [34]: xc = ["lag%d" % i for i in range(1,lag+1)]
         split = 0.66
         isplit = int(len(lagged) * split)
         xt = lagged[10:][xc]
         yt = lagged[10:]["compte"]
         X_train, y_train, X_test, y_test = xt[:isplit], yt[:isplit], xt[isplit:], yt[isplit:]

In [35]: from sklearn.ensemble import RandomForestRegressor
```

```
clr = RandomForestRegressor()
clr.fit(X_train, y_train)
```

```
Out[35]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                               max_features='auto', max_leaf_nodes=None,
                               min_impurity_decrease=0.0, min_impurity_split=None,
                               min_samples_leaf=1, min_samples_split=2,
                               min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                               oob_score=False, random_state=None, verbose=0, warm_start=False)
```

```
In [36]: from sklearn.metrics import r2_score
         r2 = r2_score(y_test.as_matrix(), clr.predict(X_test))
         r2
```

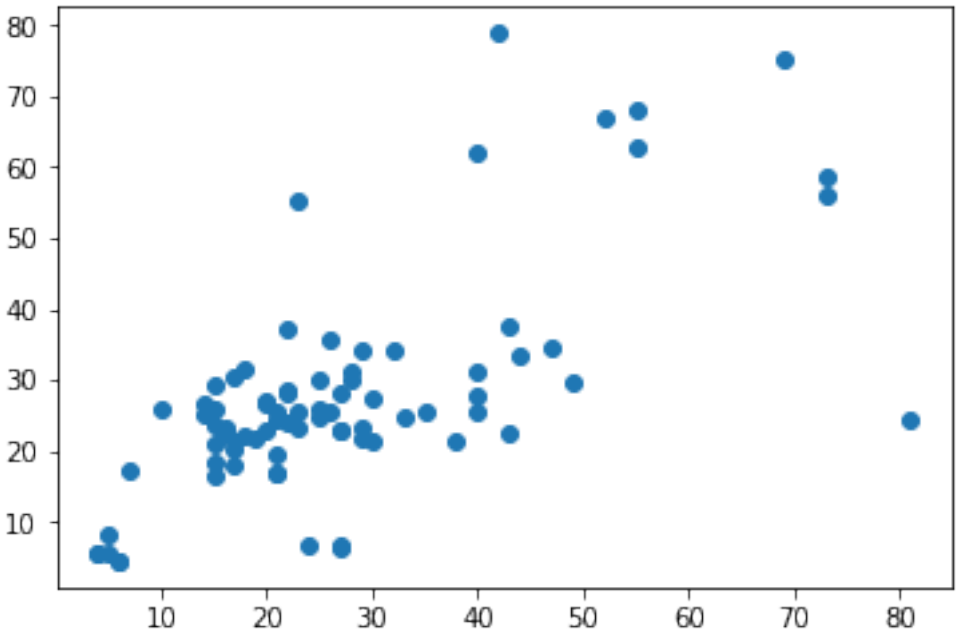
```
c:\python370_x64\lib\site-packages\ipykernel_launcher.py:2: FutureWarning: Method .as_matrix will be re
```

```
Out[36]: 0.3719371273037827
```

```
In [37]: plt.scatter(y_test.as_matrix(), clr.predict(X_test))
```

```
c:\python370_x64\lib\site-packages\ipykernel_launcher.py:1: FutureWarning: Method .as_matrix will be re
  """Entry point for launching an IPython kernel.
```

```
Out[37]: <matplotlib.collections.PathCollection at 0x284f18ee2e8>
```



1.3 Texte

On cherche à comparer une LDA avec word2vec et kmeans et les données qui sont sur ensae_teaching_cs/src/ensae_teaching_cs/data/data_web/.

In [38]:
```
from ensae_teaching_cs.data import twitter_zip
df = twitter_zip(as_df=True)
df.head(n=2).T
```

Out[38]:

	0	1
index	776066992054861825	776067660979245056
nb_user_mentions	0	0
nb_extended_entities	0	0
nb_hashtags	1	1
geo	NaN	NaN
text_hashtags	, SiJétaisPrésident	, SiJétaisPrésident
annee	2016	2016
delimit_mention	NaN	NaN
lang	fr	fr
id_str	7.76067e+17	7.76068e+17
text_mention	NaN	NaN
retweet_count	4	5
favorite_count	3	8
type_extended_entities	[]	[]
text	#SiJétaisPrésident se serait la fin du monde...	#SiJétaisPrésident je donnerai plus de vacance...
nb_user_photos	0	0
nb_urls	0	0
nb_symbols	0	0
created_at	Wed Sep 14 14:36:04 +0000 2016	Wed Sep 14 14:38:43 +0000 2016
delimit_hash	, 0, 18	, 0, 18

1.3.1 Des mots aux coordonnées - tf-idf

In [39]:
```
keep = df.text.dropna().index
```

In [40]:
```
dfnonan = df.iloc[keep, :]
dfnonan.shape
```

Out[40]: (5087, 20)

```
In [41]: from sklearn.feature_extraction.text import TfidfVectorizer
         tfidf_vectorizer = TfidfVectorizer(max_df=0.95, min_df=2, max_features=1000)
         tfidf = tfidf_vectorizer.fit_transform(dfnonan["text"])

In [42]: tfidf[:2, :]

Out[42]: <2x1000 sparse matrix of type '<class 'numpy.float64'>'
                with 21 stored elements in Compressed Sparse Row format>
```

1.3.2 LDA

```
In [43]: from sklearn.decomposition import LatentDirichletAllocation
         lda = LatentDirichletAllocation(n_components=10, max_iter=5,
                                         learning_method='online',
                                         learning_offset=50.,
                                         random_state=0)

In [44]: lda.fit(tfidf)

Out[44]: LatentDirichletAllocation(batch_size=128, doc_topic_prior=None,
                       evaluate_every=-1, learning_decay=0.7,
                       learning_method='online', learning_offset=50.0,
                       max_doc_update_iter=100, max_iter=5, mean_change_tol=0.001,
                       n_components=10, n_jobs=1, n_topics=None, perp_tol=0.1,
                       random_state=0, topic_word_prior=None,
                       total_samples=1000000.0, verbose=0)

In [45]: tf_feature_names = tfidf_vectorizer.get_feature_names()
         tf_feature_names[100:103]

Out[45]: ['avoir', 'bac', 'bah']

In [46]: lda.components_.shape

Out[46]: (10, 1000)
```

On obtient dix vecteurs qui représentent les dix vecteurs associés aux dix clusters. Chaque dimension relié au fait que le mot appartient ou non au cluster.

```
In [47]: def print_top_words(model, feature_names, n_top_words):
             for topic_idx, topic in enumerate(model.components_):
                 print("Topic #%d:" % topic_idx)
                 print(" ".join([feature_names[i]
                                 for i in topic.argsort()[:-n_top_words - 1:-1]]))
             print()

In [48]: print_top_words(lda, tf_feature_names, 10)

Topic #0:
gratuit mcdo supprimerai école soir kebab macdo kfc domicile cc
Topic #1:
macron co https de la est le il et hollande
Topic #2:
sijetaispresident je les de la et le des en pour
Topic #3:
notaires eu organiserais mets carte nouveaux journées installation cache créer
Topic #4:
```

```
sijetaispresident interdirais les je ballerines la serait serais bah de
Topic #5:
ministre de sijetaispresident la je premier mort et nommerais président
Topic #6:
cours le supprimerais jour sijetaispresident lundi samedi semaine je vendredi
Topic #7:
port interdirait démissionnerais promesses heure rendrai ballerine mes changement christineboutin
Topic #8:
seraient sijetaispresident gratuits aux les nos putain éducation nationale bonne
Topic #9:
bordel seront légaliserai putes gratuites pizza mot virerais vitesse dutreil
```

1.3.3 Clustering

```
In [49]: from sklearn.cluster import KMeans
         km = KMeans(n_clusters=10)
         km.fit(tfidf)

Out[49]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
             n_clusters=10, n_init=10, n_jobs=1, precompute_distances='auto',
             random_state=None, tol=0.0001, verbose=0)

In [50]: km.cluster_centers_.shape

Out[50]: (10, 1000)

In [51]: def print_top_words(model, feature_names, n_top_words):
             for topic_idx, topic in enumerate(model.cluster_centers_):
                 print("Topic #%d:" % topic_idx)
                 print(" ".join([feature_names[i]
                                 for i in topic.argsort()[:-n_top_words - 1:-1]]))
             print()

In [52]: print_top_words(km, tf_feature_names, 10)

Topic #0:
serais je sijetaispresident un pas ne président que de la
Topic #1:
la de sijetaispresident je et plus république aurait france ministre
Topic #2:
serait sijetaispresident la de le merde ministre ça et on
Topic #3:
le sijetaispresident je monde et de pour tout les la
Topic #4:
les sijetaispresident je et de tous seraient pour ballerines en
Topic #5:
https co macron de la le les via et sijetaispresident
Topic #6:
des sijetaispresident je de les et en la le pour
Topic #7:
une sijetaispresident je de pour et la loi en dans
Topic #8:
macron est il de hollande la pas le gauche un
Topic #9:
```

```
sijetaispresident je de en un pas que ferais au vous
```

Ah les stop words....

In [53]: