UDACITY

Return to "Data Analyst Nanodegree" in the classroom

Identify Fraud from Enron Email

REVIEW

CODE REVIEW 4

HISTORY

poi_id.py 4

```
#!/usr/bin/env python
# coding: utf-8

# ## Projeto Final - DS2

# ### Import

# In[232]:


import sys
import pickle
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
sys.path.append("../tools/")
warnings.filterwarnings("ignore")


# In[233]:


from sklearn.pipeline import Pipeline
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV,train_test_split,
from sklearn.tree import DecisionTreeClassifier
```

```
29 from sklearn.feature_selection import SelectKBest
30 from sklearn.externals import joblib
31 from sklearn.linear_model import LogisticRegression
32 from sklearn.naive_bayes import GaussianNB
33 from sklearn.grid_search import GridSearchCV
34 from sklearn.cross_validation import StratifiedShuffleSplit, train_test_split
35 from sklearn.neighbors import KNeighborsClassifier
36 from sklearn.svm import SVC
37 from sklearn.metrics import accuracy_score
38 from sklearn.preprocessing import MinMaxScaler
39 from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
40
41
42 # In[234]:
43
44
45 from time import time
46 from feature_format import featureFormat, targetFeatureSplit
47 from tester import dump_classifier_and_data, test_classifier
48 scaler = MinMaxScaler()
49
50
51 # ### Funções
52
53 # In[307]:
54
55
56 def monta_grafico(feat_x, feat_y, titulo, dicionario, cor):
```

SUGGESTION

Bom trabalho ao criar funções auxiliares! Minha sugestão é de movê-las para arquivos auxiliares.

```
57     # Criar um grafico scatter das fetures passadas no parametro
58     features = ['poi', feat_x, feat_y]
59     data = featureFormat(dicionario, features)
60
61     plt.figure(figsize=(16,7))
62
63     for point in data:
64         x = point[1]
65         y = point[2]
66         if point[0]:
67             if cor == 1:
68                 plt.scatter(x, y, color="red", marker="*")
69             else:
70                 plt.scatter(x, y, color="green", marker=".")
71         else:
72             if cor == 1:
73                 plt.scatter(x, y, color='blue', marker=".")
74             else:
75                 plt.scatter(x, y, color="orange", marker="*")
76
77
78     plt.title(titulo, fontsize=20)
79     plt.xlabel(feat_x, fontsize=18)
80     plt.ylabel(feat_y, fontsize=18)
81     pic = feat_x + feat_y + '.png'
82     plt.savefig(pic, transparent=True)
```

```
    plt.show()


def monta_feature(features_list):
    features_list = ['poi',
                     'salary',
                     'deferral_payments',
                     'total_payments',
                     'loan_advances',
                     'bonus',
                     'restricted_stock_deferred',
                     'deferred_income',
                     'total_stock_value',
                     'expenses',
                     'exercised_stock_options',
                     'other',
                     'long_term_incentive',
                     'restricted_stock',
                     'director_fees',
                     'to_messages',
                     'from_poi_to_this_person',
                     'from_messages',
                     'from_this_person_to_poi',
                     'shared_receipt_with_poi']
    return features_list

def nova_feature(dataset, features_list):
    nova_feature = ["fraction_from_poi_email", "fraction_to_poi_email"]
    num_features = ["from_poi_to_this_person", "from_this_person_to_poi"]
    den_features = ["to_messages", "from_messages"]

    for x in dataset:
        data = dataset[x]

        for i, feature in enumerate(nova_feature):
            if data["poi"]:
                data[feature] = 'NaN'
            else:
                message_poi = data[num_features[i]]
                messages_all = data[den_features[i]]
                fracao_messages = calcula_fracao(message_poi, messages_all)
                data[feature] = fracao_messages

    return features_list + nova_feature


def testa_nova_feature(dataset, x, nova_feature):
    num_features = ["from_poi_to_this_person", "from_this_person_to_poi"]
    den_features = ["to_messages", "from_messages"]

    print x, "\n- {} = {:.4f} ({} / {})\n- {} = {:.4f} ({} / {})\n".format(nova_featu


def pipeline_classificador(tipo, kbest, f_list):
    # Contruir um pipeline e tune parameters via GridSearchCV

    data = featureFormat(my_dataset, f_list, sort_keys=True)
    labels, features = targetFeatureSplit(data)

    # Usando o stratified shuffle split cross validation devido ao tamanho dos conjun
    stratified split cross validation = StratifiedShuffleSplit(labels, 500, test size
```

```
144
145     # Build pipeline
146     kbest = SelectKBest(k=kbest)
147     scaler = MinMaxScaler()
148     classifier = escolher_classificador(tipo)
149     pipeline = Pipeline(steps=[('minmax_scaler', scaler), ('feature_selection', kbest
150
151     # Set parameters for random forest
152     parameters = []
153     if tipo == 'randomforest':
154         parameters = dict(randomforest__n_estimators=[25, 50],
155                           randomforest__min_samples_split=[2, 3, 4],
156                           randomforest__criterion=['gini', 'entropy'])
157     if tipo == 'logistic_regression':
158         parameters = dict(logistic_regression__class_weight=['balanced'],
159                           logistic_regression__solver=['liblinear'],
160                           logistic_regression__C=range(1, 5),
161                           logistic_regression__random_state=42)
162     if tipo == 'decisiontree':
163         parameters = dict(decisiontree__min_samples_leaf=range(1, 5),
```

SUGGESTION

Aqui seria interessante também tunar o parâmetro `k` do `feature_selection`!

```
164                           decisiontree__max_depth=range(1, 5),
165                           decisiontree__class_weight=['balanced'],
166                           decisiontree__criterion=['gini', 'entropy'])
167
168     # Get optimized parameters for F1-scoring metrics
169     cv = GridSearchCV(pipeline, param_grid=parameters, scoring='f1', cv=stratified_sp
```

AWESOME

Ótima implementação!

```
170     t0 = time()
171     cv.fit(features, labels)
172     print 'Tuning Classifier: %r' % round(time() - t0, 3)
173
174     return cv
175
176 def melhor_classificador(aux):
177     # Função para escolher um tipo de classificador
178     return {
179         'randomforest': RandomForestClassifier(),
180         'decisiontree': DecisionTreeClassifier(),
181         'logistic_regression': LogisticRegression(),
182         'gaussiannb': GaussianNB()
183     }.get(aux)
184
185 def calcula_fracao(message_poi, messages_all):
186     calc_fracao = 0.
187     if message_poi != "NaN" and messages_all != "NaN":
188         calc_fracao = float(message_poi) / messages_all
189     return calc_fracao
```

```
190
191 def sumariza_valores(dataset):
192     df_list = []
193     for key, y in dataset.items():
194         df_list.append(y)
195
196     df = pd.DataFrame(df_list, columns = dataset.items()[0][1].keys())
197
198     for i in df.columns:
199         df[i][df[i].apply(lambda i: True if str(i) == "NaN" else False)]=None
200
201     df = df.convert_objects(convert_numeric=True)
202     df.info()
203
204 def conta_poi(dataset):
205     poi_count = 0
206     for key, value in dataset.items():
207         if value['poi']:
208             poi_count += 1
209     return poi_count
210
211 def accuracy(new_features,features_list):
212     #Feature List
213     features_list = monta_feature(features_list)
214     if new_features == False:
215         print features_list
216
217     else:
218         features_list = nova_feature(data_dict_woo, features_list)
219         print ""
220         print features_list
221         print "Testando novos features adicionados:\n"
222         testa_nova_feature(data_dict_woo, "DIETRICH JANET R", features_list[-2:])
223
224     # Extraindo as features e os labels do conjunto de dados
225     data = featureFormat(my_dataset, features_list, sort_keys = True)
226     labels, features = targetFeatureSplit(data)
227
228     features_train, features_test, labels_train, labels_test =     train_test_split(f
229     print ""
230
231     # Criando Min/Max Scaler
232     from sklearn import preprocessing
233     scaler = preprocessing.MinMaxScaler()
234     # Scale Features
235     features = scaler.fit_transform(features)
236
237     skbest = SelectKBest(k=10)  # try best value to fit
238     sk_trans = skbest.fit_transform(features_train, labels_train)
239     indices = skbest.get_support(True)
240
241     print "="*10,"skbest.scores_","="*10
242     print skbest.scores_
243     print "="*10, "="*(len("skbest.scores_")-2),"="*10
244     print ""
245
246     print "="*10,"features - score","="*10
247     for index in indices:
248         print 'features: %s score: %f' % (features_list[index + 1], skbest.scores_[in
249
250     print "="*10, "="*(len('features: %s score: %f')-2),"="*10
```

```
    print ""

    #print "GaussianNB"
    # GaussianNB
    clf = GaussianNB()
    clf.fit(features_train, labels_train)
    prediction = clf.predict(features_test)
    print "Accuracy GaussianNB  = {:.5f}".format(accuracy_score(prediction, labels_te

    #print "KNeighborsClassifier"
    # KNeighborsClassifier
    clf = KNeighborsClassifier()
    clf = KNeighborsClassifier(algorithm = 'auto',leaf_size = 20,n_neighbors = 3,weig
    clf.fit(features_train, labels_train)
    prediction = clf.predict(features_test)
    print "Accuracy KNeighborsClassifier  = {:.5f}".format(accuracy_score(prediction,

    #print "SVC"
    # SVC
    clf = SVC(kernel = 'linear',max_iter = 10000,random_state = 42)
    clf.fit(features_train, labels_train)
    prediction = clf.predict(features_test)
    print "Accuracy SVC = {:.5f}".format(accuracy_score(prediction, labels_test))

    #print "AdaBoostClassifier"
    clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1, min_samples_leaf=2,
                             n_estimators=50, learning_rate=.8)
    clf.fit(features_train, labels_train)
    prediction = clf.predict(features_test)
    print "Accuracy AdaBoostClassifier = {:.5f}".format(accuracy_score(prediction, la

def adaboost_kbest(kbest_value, new_features,features_list):
    #Feature List
    features_list = monta_feature(features_list)
    if new_features == False:
        print "Default Features"
#        print features_list

    else:
        features_list = nova_feature(data_dict_woo, features_list)
        print "New Features"
#        print features_list
#        print "Testando novos features adicionados:\n"
#        testa_nova_feature(data_dict_woo, "DIETRICH JANET R", features_list[-2:])

    # Extraindo as features e os labels do conjunto de dados
    data = featureFormat(my_dataset, features_list, sort_keys = True)
    labels, features = targetFeatureSplit(data)

    features_train, features_test, labels_train, labels_test =     train_test_split(f
    print ""

    # Criando Min/Max Scaler
    from sklearn import preprocessing
    scaler = preprocessing.MinMaxScaler()
    # Scale Features
    features = scaler.fit_transform(features)

    skbest = SelectKBest(k=kbest_value)  # try best value to fit
    sk_trans = skbest.fit_transform(features_train, labels_train)
    indices = skbest.get_support(True)
```

```
    clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1, min_samples_leaf=3,
                         n_estimators=50, learning_rate=.95)

    # Validate model precision, recall and F1-score
    test_classifier(clf, my_dataset, features_list)



# ### Task 1: Select what features you'll use.
# - features_list is a list of strings, each of which is a feature name.
# - The first feature must be "poi".

# In[236]:


features_list = []
features_list = monta_feature(features_list)


# In[237]:


print features_list


# In[238]:


#Feature List
#features_list = ['poi',
#                 'salary',
#                 'deferral_payments',
#                 'total_payments',
#                 'loan_advances',
#                 'bonus',
#                 'restricted_stock_deferred',
#                 'deferred_income',
#                 'total_stock_value',
#                 'expenses',
#                 'exercised_stock_options',
#                 'other',
#                 'long_term_incentive',
#                 'restricted_stock',
#                 'director_fees',
#                 'to_messages',
#                 'from_poi_to_this_person',
#                 'from_messages',
#                 'from_this_person_to_poi',
#                 'shared_receipt_with_poi'] # You will need to use more features


# In[239]:


# Carregando o conjunto de dados
with open("final_project_dataset.pkl", "r") as data_file:
#    data_dict = pickle.load(data_file)
    data_dict = pickle.load(open("final_project_dataset.pkl", "r"))
```

```
373
374 # In[240]:
375
376
377 print "\nQuantidade total de registros: {}\nQuantidade total de features: {}".format(
378 print "Quantidade de POI's: {}".format(conta_poi(data_dict))
379 print "Quantidade de não POI's: {}".format(len(data_dict) - conta_poi(data_dict))
380
381
382 # In[241]:
383
384
385 df_enron = pd.DataFrame.from_dict(data_dict, orient = 'index')
386
387
388 # In[242]:
389
390
391 df_enron.head()
392
393
394 # In[243]:
395
396
397 df_enron.sample(5)
398
399
400 # In[244]:
401
402
403 df_enron.tail()
404
405
406 # In[245]:
407
408
409 df_enron.describe().transpose()
410
411
412 # In[246]:
413
414
415 monta_grafico("salary", "bonus", "Primeira Analise",data_dict,1)
416
417
418 # ### Analise Exploratória
419 # - Visulamente já é possível ver que muitas informações estão faltando
420 # - Isso fica bem evidente quando analisamos por código
421 # - Foi identificado o outliers TOTAL, que nos levou a investigar possíveis pois que
422
423 # In[247]:
424
425
426 print "Dados: HAUG DAVID L:\n\n{}".format(data_dict["TOTAL"])
427 print "Dados: LOCKHART EUGENE E:\n\n{}".format(data_dict["LOCKHART EUGENE E"])
428 print "Dados: THE TRAVEL AGENCY IN THE PARK:\n\n{}".format(data_dict["THE TRAVEL AGEN
```

AWESOME

Bom trabalho ao identificar e remover estes outliers!

Bom trabalho ao identificar e remover estes outliers!

```
# ### Task 2: Remove outliers

# In[248]:

# Removendo outliers
with open("final_project_dataset.pkl", "r") as data_file:
    data_dict_woo = pickle.load(data_file)

print "Quantidade total de registros Com outliers: {}\n".format(len(data_dict_woo))

data_dict_woo.pop('TOTAL', None) #Não é um funcionário
data_dict_woo.pop('LOCKHART EUGENE E', None) #Não é um funcionário
data_dict_woo.pop('THE TRAVEL AGENCY IN THE PARK', None) #Não é um funcionário

print "Quantidade total de registros Sem outliers: {}\n".format(len(data_dict_woo))


# In[249]:


monta_grafico("salary", "bonus", "Com Outliers",data_dict,1)
monta_grafico("salary", "bonus", "Sem Outliers",data_dict_woo,2)


# In[250]:


df_enron_woo = pd.DataFrame.from_dict(data_dict_woo, orient = 'index')
df_enron_woo.describe().transpose()


# In[251]:


df_enron_woo.head(20)


# In[252]:


sumariza_valores(data_dict_woo)


# In[253]:


#df_graph_enron = df_enron_woo.copy()
df_graph_enron = df_enron_woo.describe().transpose()


# In[254]:


df_graph_enron.info()
```

```
# In[ ]:



# In[255]:


# Processo de criação de coluna com a diferença entre o total de registros unicos
#e registros duplicados/Não atribuídos

nan_dupl_col = []


def nan_dupl(reg):
    qty_nan_dupl = reg['count'] - reg['unique']
    nan_dupl_col.append(qty_nan_dupl)

df_graph_enron.apply(nan_dupl, axis=1)
df_graph_enron['nan_dupl'] = nan_dupl_col


# In[256]:


df_graph_enron.drop(['count', 'top', 'freq'], axis=1, inplace=True)

df_graph_enron.head()


# In[257]:


df_graph_enron.info()


# In[258]:


df_graph_enron.plot.bar()


# In[259]:


labels_graph = list(df_graph_enron.index.values)
print labels_graph


# In[260]:



bar_1 = df_graph_enron['unique']
bar_2 = df_graph_enron['nan_dupl']
x_pos = np.arange(len(bar_1))

plt2 = plt

plt.figure(figsize=(15,8))
```

```
549
550 #plt.rcParams["figure.figsize"] = [15,8]
551 #plt.rcParams["legend.frameon"] = True
552 #plt.rcParams["legend.handletextpad"] = 1
553 #plt.rcParams["legend.borderaxespad"] = 60
554
555 first_bar = plt.bar(x_pos, bar_1, 0.5, color='green')
556 second_bar = plt.bar(x_pos, bar_2, 0.5, color='skyblue', bottom=bar_1)
557 plt.title('Quantity Unique & Not a Number or Duplicated X Feature', fontsize=22)
558 plt.xlabel('Features', fontsize=18)
559 plt.ylabel('Quantity', fontsize=18)
560
561 # Definir posição e labels no eixo X
562 plt.xticks(x_pos, (labels_graph), rotation=90)
563
564 def autolabel(rects, xpos='center'):
565     """
566     Attach a text label above each bar in *rects*, displaying its height.
567
568     *xpos* indicates which side to place the text w.r.t. the center of
569     the bar. It can be one of the following {'center', 'right', 'left'}.
570     """
571
572     xpos = xpos.lower()  # normalize the case of the parameter
573     ha = {'center': 'center', 'right': 'left', 'left': 'right'}
574     offset = {'center': 0.5, 'right': 0.57, 'left': 0.43}  # x_txt = x + w*off
575
576
577     if rects == first_bar:
578         for rect in rects:
579             height = rect.get_height()
580             plt.text(rect.get_x() + rect.get_width()*offset[xpos], 1.00*height,
581                     '{}'.format(height), ha=ha[xpos], va='bottom')
582     else:
583         for rect in rects:
584             height = rect.get_height()
585             plt.text(rect.get_x() + rect.get_width()*offset[xpos], 1.00*138.5,
586                     '{}'.format(height), ha=ha[xpos], va='bottom')
587
588
589 autolabel(first_bar, "center")
590 autolabel(second_bar, "center")
591
592 plt.legend(labels=['Unique','Not a Number or Duplicated'], loc=8, borderaxespad = 47
593
594 plt.show()
595
596
597 # In[ ]:
598
599
600
601
602
603 # ### Task 3: Create new feature(s)
604
605 # In[283]:
606
607
608 # Salvando o conjunto de dados
609 #df_enron_woo = pd.DataFrame.from_dict(data_dict_woo, orient = 'index')
```

```
#df_enron_woo.replace(to_replace='NaN', value=0.0, inplace=True)
my_dataset = data_dict_woo
#my_dataset = df_enron_woo.to_dict('index')


# In[284]:


#Feature List
features_list = ['poi',
                 'salary',
                 'deferral_payments',
                 'total_payments',
                 'loan_advances',
                 'bonus',
                 'restricted_stock_deferred',
                 'deferred_income',
                 'total_stock_value',
                 'expenses',
                 'exercised_stock_options',
                 'other',
                 'long_term_incentive',
                 'restricted_stock',
                 'director_fees',
                 'to_messages',
                 'from_poi_to_this_person',
                 'from_messages',
                 'from_this_person_to_poi',
                 'shared_receipt_with_poi'] # You will need to use more features

features_list = nova_feature(data_dict_woo, features_list)
print features_list


# In[285]:


print "Testando novos features adicionados:\n"
testa_nova_feature(data_dict_woo, "DIETRICH JANET R", features_list[-2:])


# In[286]:


# Extraindo as features e os labels do conjunto de dados
data = featureFormat(my_dataset, features_list, sort_keys = True)
labels, features = targetFeatureSplit(data)


# In[287]:


# Criando Min/Max Scaler
from sklearn import preprocessing
scaler = preprocessing.MinMaxScaler()
# Scale Features
features = scaler.fit_transform(features)


# ### Task 4: Try a varity of classifiers
# - Please name your classifier clf for easy export below.
```

```
# - Note that if you want to do PCA or other multi-stage operations,
# - you'll need to use Pipelines. For more info: http://scikit-learn.org/stable/modul

# In[288]:


#features_train, features_test, labels_train, labels_test = \
#    train_test_split(features, labels, test_size=0.3, random_state=42)


# In[289]:


#skbest = SelectKBest(k=10)  # try best value to fit
#sk_trans = skbest.fit_transform(features_train, labels_train)
#indices = skbest.get_support(True)
#print skbest.scores_


# In[290]:


#for index in indices:
#     print 'features: %s score: %f' % (features_list[index + 1], skbest.scores_[inde
#print ""


# In[291]:


##print "GaussianNB"
## GaussianNB
#clf = GaussianNB()
#clf.fit(features_train, labels_train)
#prediction = clf.predict(features_test)
#print ("Accuracy GaussianNB =", accuracy_score(prediction0, labels_test))


# In[292]:


#print "KNeighborsClassifier"
# KNeighborsClassifier
#clf = KNeighborsClassifier()
#clf = KNeighborsClassifier(algorithm = 'auto',leaf_size = 20,n_neighbors = 3,weights
#clf.fit(features_train, labels_train)
#prediction = clf.predict(features_test)
#print "Accuracy KNeighborsClassifier =", accuracy_score(prediction, labels_test)


# In[293]:


#print "SVC"
# SVC
#clf = SVC(kernel = 'linear',max_iter = 10000,random_state = 42)
#clf.fit(features_train, labels_train)
#prediction = clf.predict(features_test)
#print "Accuracy SVC =", accuracy_score(prediction, labels_test)

```

```
# In[294]:


#print "AdaBoostClassifier"
#clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1, min_samples_leaf=2, cla
#                         n_estimators=50, learning_rate=.8)
#clf.fit(features_train, labels_train)
#prediction = clf.predict(features_test)
#print "Accuracy AdaBoostClassifier =", accuracy_score(prediction, labels_test)


# In[295]:


accuracy(False, features_list)


# In[296]:


accuracy(True, features_list)


# In[297]:


data = {"Algorithms":["GaussianNB","KNeighborsClassifier",
"SVC","AdaBoostClassifier",
"GaussianNB","KNeighborsClassifier",
"SVC","AdaBoostClassifier"                    ],
"New Features":["N","N","N","N","S","S","S","S"],
"Accuracy":[0.88372,0.90698,0.88372,0.81395,0.88372,0.90698,0.88372,0.95349],
}
#Accuracy GaussianNB  = 0.88372
#Accuracy KNeighborsClassifier  = 0.90698
#Accuracy SVC = 0.88372
#Accuracy AdaBoostClassifier = 0.81395
#Accuracy GaussianNB  = 0.88372
#Accuracy KNeighborsClassifier  = 0.90698
#Accuracy SVC = 0.88372
#Accuracy AdaBoostClassifier = 0.95349
algorithms = pd.DataFrame(data, columns = ["Algorithms", "New Features", "Accuracy",]
algorithms


# ### Task 5: Tune your classifier
#     Tune your classifierto achieve better than .3 precision and recall
#     using our testing script. Check the tester.py script in the final project
#     folder for details on the evaluation method, especially the test_classifier
#     function. Because of the small size of the dataset, the script uses
#     stratified shuffle split cross validation. For more info:

# In[212]:


# Testar os classificadores
#print '\n'
#print '########## Testar and Tunning Classifiers ##########'
# See "pipeline_classificador" for MinMaxScaling, SelectKBest and Logistic Regression

# Classifiers tested but not using - Logistic Regression, RandomForestClassifier, Dec
```

```
793
794 #cross_val = pipeline_classificador('randomforest',9, features_list)
795 #print 'Melhores Parametros: ', cross_val.best_params_
796 #clf = cross_val.best_estimator_
797
798
799
800 # In[ ]:
801
802
803
804
805
806 # In[281]:
807
808
809 clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1, min_samples_leaf=2, clas
810                          n_estimators=50, learning_rate=.8)
811
812 #clf.fit(features_train, labels_train)
813 #prediction = clf.predict(features_test)
814 #print "Accuracy AdaBoostClassifier =", accuracy_score(prediction, labels_test)
815
816
817 # Validate model precision, recall and F1-score
818 test_classifier(clf, my_dataset, features_list)
819
820
821 # In[308]:
822
823
824 for k in range(1,11):
825     print "========== kbest = ", k, "="*10, "Ini"
826
827     t0 = time()
828     adaboost_kbest(k, True, features_list)
829
830     print "tempo de treinamento:", round(time()-t0, 3), "s"
831     print "========== kbest = ", k, "="*10,"Fim"
832
833
834
835
836 # In[309]:
837
838
839 for k in range(1,11):
840     print "========== kbest = ", k, "="*10, "Ini"
841     t0 = time()
842     adaboost_kbest(k, False, features_list)
843     print "tempo de treinamento:", round(time()-t0, 3), "s"
844     print "========== kbest = ", k, "="*10,"Fim"
845
846
847 # ### Task 6: Dump your classifier, dataset, and features_list
848 # Dump your classifier, dataset, and features_list so anyone can check your results.
849 # You do not need to change anything below, but make sure that the version of
850 # poi_id.py that you submit can be run on its own and generates the necessary .pkl
851 # files for validating your results.
852
853 # In[103]:
```

```
854
855
856 dump_classifier_and_data(clf, my_dataset, features_list)
857
858
859 # ### Refrências
860 # https://datascience.stackexchange.com/questions/13410/parameters-in-gridsearchcv-in
861 #
862 # http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier
863 #
864 # https://stackoverflow.com/questions/45444953/parameter-values-for-parameter-n-estima
865 #
866 # http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier
867 #
868 # http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedS
869 #
870 # https://stackoverflow.com/questions/45969390/difference-between-stratifiedkfold-and
871 #
872 # https://www.featurelabs.com/blog/feature-engineering-vs-feature-selection/
873 #
874 # http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overf
875 #
876 # http://scikit-learn.org/stable/auto_examples/feature_selection/plot_rfe_with_cross_v
877 #
878 # https://en.wikipedia.org/wiki/Enron
879 #
880 # http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
881 #
882 # http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.ht
883 #
884 # http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler
885 #
886 # http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
887 #
888 # http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBe
889 #
890 # http://scikit-learn.org/stable/auto_examples/plot_compare_reduction.html#sphx-glr-au
891 #
892 # https://en.wikipedia.org/wiki/Precision_and_recall
893 #
894 # http://scikit-learn.org/stable/tutorial/statistical_inference/putting_together.html
895 #
896 # https://datascience.stackexchange.com/questions/21877/how-to-use-the-output-of-grids
897 #
898 # https://stats.stackexchange.com/questions/62621/recall-and-precision-in-classificati
899 #
900 # https://stackoverflow.com/questions/49147774/what-is-random-state-in-sklearn-model-s
901 #
902 # @Jefferson Aparecido Rodrigues ( Aluno Udacity que me deu umas dicas, me destravando
903
904 # In[ ]:
905
906
907
908
909
910 # In[ ]:
911
912
913
914
```

```
914
915
```

RETURN TO PATH