experimento 1

April 15, 2019

In [1]:
```
import pandas as pd
import numpy as np
from os import path
from sklearn.externals.joblib import Parallel, delayed
```

In [2]:
```
data = pd.read_csv('/Users/felipecordeiro/IdeaProjects/mobility-analysis/scripts/data.c
y = pd.read_csv('/Users/felipecordeiro/IdeaProjects/mobility-analysis/scripts/labels.cs
```

In [3]:
```
X = np.squeeze(data.values)
y = np.squeeze(y.values)
```

In [4]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, co
from sklearn import tree, svm, neighbors, model_selection
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
import numpy as np
```

Nao rodei experimentos para sklearn.naive_bayes.ComplementNB - está falhando o import

In [5]:
```
from sklearn.naive_bayes import ComplementNB
models = []

cnb_clf = ComplementNB()
parameters = {
    "complementnb__alpha": [0, 1, 3, 5],
    "complementnb__norm": [True, False],

}
cnb_clf.random_state = 42
np.random.seed(42)
models.append(("naive_bayes.ComplementNB", cnb_clf, parameters))
```

In [ ]:

In [6]:
```
'''
random_forest_clf = RandomForestClassifier()
random_forest_clf.random_state = seed
parameters = {
    "randomforestclassifier__n_estimators":[10, 20, 30],
    "randomforestclassifier__max_depth": [None, 5, 10, 15, 20]
}
models.append(("RandomForestClassifier", random_forest_clf, parameters))



mnb_clf = MultinomialNB()
parameters = {
    "multinomialnb__alpha": [0, 1, 3, 5, 10]
}
mnb_clf.random_state = seed
models.append(("naive_bayes.MultinomialNB", mnb_clf, parameters))



decision_tree_clf = tree.DecisionTreeClassifier()
decision_tree_clf.random_state = seed
parameters = {
    "decisiontreeclassifier__max_depth": [None, 2, 5, 8, 10, 15]
}
models.append(("decision tree", decision_tree_clf, parameters))


logistic_regression_clf = LogisticRegression()
logistic_regression_clf.random_state = seed
parameters = {
    "logisticregression__penalty": ['l1', 'l2'],
    "logisticregression__C": [0.01, 0.1, 1, 5, 10, 15, 20]
}
models.append(("logistic regression", logistic_regression_clf, parameters))


svm_clf = svm.SVC()
svm_clf.random_state = seed
parameters = {'svc__kernel':('linear', 'rbf'), 'svc__C':[0.1, 1, 5, 10, 15, 20]}
models.append(("svm", svm_clf, parameters))



mlp_clf = MLPClassifier()
mlp_clf.random_state = seed

parameters = {'mlpclassifier__solver': ['lbfgs'],
```

```
                    'mlpclassifier__activation':['relu'],
                    'mlpclassifier__max_iter': [2000],
                    'mlpclassifier__alpha': 10.0 ** -np.arange(1, 10),
                    'mlpclassifier__tol':[1e-3],
                    'mlpclassifier__hidden_layer_sizes':np.arange(10, 15)}

        models.append(("mlp", mlp_clf, parameters))



        '''

        # knn_clf = neighbors.KNeighborsClassifier()
        # knn_clf.random_state = seed
        # parameters = {
        #     "kneighborsclassifier__n_neighbors": [1, 3, 5, 7, 9],
        #     "kneighborsclassifier__metric":["cosine", "euclidean"]
        # }
        # models.append(("knn", knn_clf, parameters))
```

Out[6]: '\nrandom_forest_clf = RandomForestClassifier()\nrandom_forest_clf.random_state = seed'

```
In [7]: scoring = {'accuracy': make_scorer(accuracy_score),
                            'precision_micro': make_scorer(precision_score, pos_label=None, a
                            'recall_micro': make_scorer(recall_score, pos_label=None, average
                            'f1_score_micro': make_scorer(f1_score, pos_label=None, average=
                            'precision_macro': make_scorer(precision_score, pos_label=None, a
                            'recall_macro': make_scorer(recall_score, pos_label=None, average
                            'f1_score_macro': make_scorer(f1_score, pos_label=None, average=
```

```
In [8]: from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.pipeline import Pipeline
        from sklearn.model_selection import GridSearchCV
        from sklearn.model_selection import fit_grid_point
        from sklearn.pipeline import make_pipeline
```

```
In [9]: from sklearn import model_selection
        skf = model_selection.StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
        skf
```

Out[9]: StratifiedKFold(n_splits=10, random_state=42, shuffle=True)

```
In [10]: results = pd.DataFrame()
         for name, model, parameter in models:
             print('modelo: ', name)
             print(parameter)

             tfidf=TfidfVectorizer(sublinear_tf=True, use_idf=True)
             pipe_model = make_pipeline(tfidf, model)
```

```
            gs = GridSearchCV(estimator=pipe_model, param_grid=parameter, scoring=scoring, re
            gs.fit(X, y)
            print("Best Estimator: \n{}\n".format(gs.best_estimator_))
            print("Best Parameters: \n{}\n".format(gs.best_params_))
            print("Best Test Score: \n{}\n".format(gs.best_score_))
            d = gs.cv_results_
            d['metodo'] = name
            df = pd.DataFrame.from_dict(d)
            results = results.append(df, ignore_index=True)
            print(results)

            results.to_csv('resultados.csv', sep=';', decimal=',')
            print('_____________________________________________________')
```

```
modelo:  naive_bayes.ComplementNB
{'complementnb__alpha': [0, 1, 3, 5], 'complementnb__norm': [True, False]}
Fitting 10 folds for each of 8 candidates, totalling 80 fits


[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done  10 tasks      | elapsed:   48.0s
[Parallel(n_jobs=4)]: Done  64 tasks      | elapsed:  3.9min
[Parallel(n_jobs=4)]: Done  80 out of  80 | elapsed:  4.9min finished


Best Estimator:
Pipeline(memory=None,
     steps=[('tfidfvectorizer', TfidfVectorizer(analyzer='word', binary=False, decode_error='st
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 1), norm='l2', preprocessor=None, smooth...bulary=None)), ('complementn

Best Parameters:
{'complementnb__alpha': 1, 'complementnb__norm': True}

Best Test Score:
0.9589531035865896

   mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0       1.674078      0.044038         1.314617        0.191792
1       1.864232      0.343803         1.488909        0.335946
2       1.448802      0.065603         1.223010        0.081110
3       1.604972      0.153314         1.197670        0.082874
4       1.471280      0.051409         1.154460        0.024375
5       1.448652      0.025417         1.131163        0.018919
6       1.567797      0.108409         1.243982        0.065907
7       1.706799      0.039831         1.342340        0.048329
```

	param_complementnb__alpha	param_complementnb__norm	\
0	0	True	
1	0	False	
2	1	True	
3	1	False	
4	3	True	
5	3	False	
6	5	True	
7	5	False	

	params	split0_test_accuracy	\
0	{'complementnb__alpha': 0, 'complementnb__norm...	0.471011	
1	{'complementnb__alpha': 0, 'complementnb__norm...	0.949541	
2	{'complementnb__alpha': 1, 'complementnb__norm...	0.954887	
3	{'complementnb__alpha': 1, 'complementnb__norm...	0.951211	
4	{'complementnb__alpha': 3, 'complementnb__norm...	0.953383	
5	{'complementnb__alpha': 3, 'complementnb__norm...	0.947034	
6	{'complementnb__alpha': 5, 'complementnb__norm...	0.953049	
7	{'complementnb__alpha': 5, 'complementnb__norm...	0.946867	

	split1_test_accuracy	split2_test_accuracy	...	\
0	0.481370	0.453634	...	
1	0.952047	0.950710	...	
2	0.955890	0.959566	...	
3	0.951211	0.954219	...	
4	0.953551	0.956224	...	
5	0.947201	0.952214	...	
6	0.951044	0.953049	...	
7	0.946366	0.952047	...	

	split3_train_f1_score_macro	split4_train_f1_score_macro	\
0	0.556831	0.576543	
1	0.899845	0.900391	
2	0.871228	0.870995	
3	0.859688	0.860959	
4	0.842413	0.841793	
5	0.838184	0.837813	
6	0.830027	0.830009	
7	0.829530	0.829404	

	split5_train_f1_score_macro	split6_train_f1_score_macro	\
0	0.567005	0.584906	
1	0.898857	0.899973	
2	0.871056	0.871384	
3	0.859340	0.860245	
4	0.841600	0.842769	
5	0.837055	0.837473	
6	0.828607	0.828191	

```
7                    0.827686                    0.827960

   split7_train_f1_score_macro  split8_train_f1_score_macro  \
0                     0.564672                     0.575933
1                     0.901111                     0.898736
2                     0.872159                     0.869824
3                     0.860117                     0.858796
4                     0.842633                     0.840245
5                     0.839946                     0.837584
6                     0.828528                     0.828075
7                     0.828658                     0.827613

   split9_train_f1_score_macro  mean_train_f1_score_macro  \
0                     0.577309                   0.570292
1                     0.898424                   0.899771
2                     0.873103                   0.871913
3                     0.859311                   0.860343
4                     0.841243                   0.842278
5                     0.839166                   0.838792
6                     0.828520                   0.829054
7                     0.830021                   0.829252

   std_train_f1_score_macro                    metodo
0                  0.008644  naive_bayes.ComplementNB
1                  0.000928  naive_bayes.ComplementNB
2                  0.001437  naive_bayes.ComplementNB
3                  0.001035  naive_bayes.ComplementNB
4                  0.001149  naive_bayes.ComplementNB
5                  0.001303  naive_bayes.ComplementNB
6                  0.000791  naive_bayes.ComplementNB
7                  0.001475  naive_bayes.ComplementNB

[8 rows x 183 columns]
----------------------------------------------------------
```

In [ ]:

In [ ]:

In [ ]: