MACHINE LEARNING

Estudo de um dataset sobre dados biométricos de alunos em situação de exame com scikit-learn

André Pereira, Carlos Lemos, João Barreira e Rafael Costa

Aprendizagem e Extração de Conhecimento – Sistemas Inteligentes
2018/2019 – MEI/MIEI – Universidade do Minho

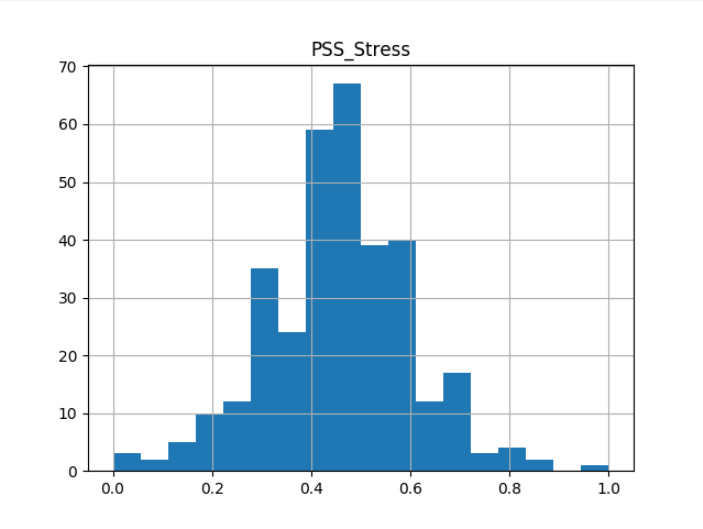
Data Acquisition

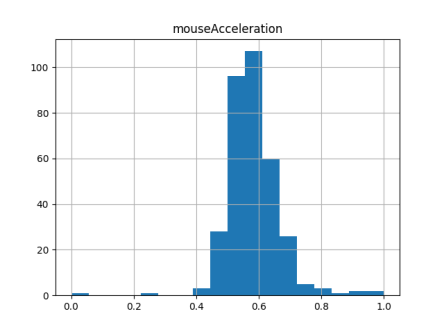
- Dados biométricos de alunos universitários em exame
- Variáveis relativas ao uso do rato

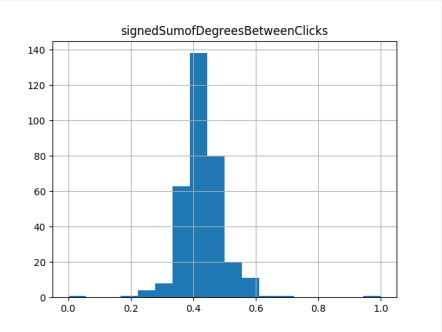
Dataset_MousePSS

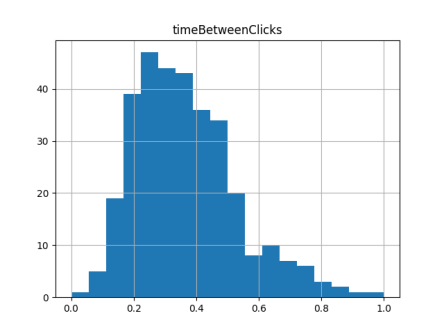
Clicks	clickDurations	distanceBetweenClicks	distanceDuringClicks	distancePointerToLineBetwee
03448	131.36937	908.45656	16.60746	12372
79938	106.27778	509.86356	1	3776
53193	130.71901	756.85671	8.02669	6642
55602	194.06593	394.32331	62.93412	867
42689	239.83019	905.85181	64.23758	8264
33043	110.38393	689.66592	11.78254	8245
83791	191.44138	515.35814	43.44354	4912
42302	181.42938	412.82752	62.68363	2880
43881	147.45029	459.20856	13.63856	2994
96973	114.66667	1940.54839	NaN	22492
59303	126.65625	757.62559	46.34985	5359
00655	492.29268	579.6267	116.33102	5604
72808	177.15239	245.58019	27.46356	1620
47877	225.57237	664.38444	54.73228	8094
41041	206.21429	392.63097	11.49548	2867

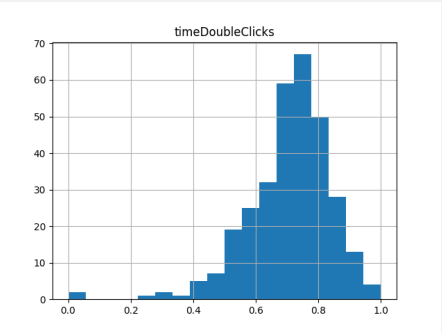
Data Visualization











Data Preprocessing

- Missing Data Filtering

```
data = data.fillna(data.median())
```

- Feature Selection

```
selector = SelectKBest(f_classif, k=5)
selector.fit(data, target)
cols = selector.get_support(indices=True)
cols_names = list(data.columns[cols])

for idx, (ci, cn) in enumerate(zip(cols, cols_names)):
    print("*" * (len(cols) - idx) + " " * idx, ci, cn)

data = data[cols_names]
```

- Normalization/Standardization

```
normalizer = preprocessing.Normalizer(norm='l1')
values_normalized = normalizer.transform(data.values)
data = pd.DataFrame(values_normalized, columns=data.columns)
```

```
robust_scaler = preprocessing.RobustScaler()
values_standardized = robust_scaler.fit_transform(data.values)
data = pd.DataFrame(values_standardized, columns=data.columns)
```

Model Selection, Training and Validation

- Para vários modelos:
  - K-Fold Cross Validation
  - Cálculo de resultado

```
clf_models = [
    KNeighborsClassifier(),
    SVC(),
    GaussianProcessClassifier(),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    MLPClassifier(),
    AdaBoostClassifier(),
    GaussianNB()
]
```

```
for name, clf in zip(clf_names, clf_models):
    scores = cross_val_score(clf, data, target, cv=5)
    print(name, "Accuracy: %0.6f (+/- %0.6f)" % (scores.mean(), scores.std() * 2))
```

Model Selection, Training and Validation

- Seleção do melhor modelo

```
***** 3 clickDurations
****  4 distanceBetweenClicks
***   6 distancePointerToLineBetweenClicks
**    7 excessOfDistanceBetweenClicks
*     10 signedSumofDegreesBetweenClicks
 ### Normalization ###
Nearest Neighbors Accuracy: 0.039258 (+/- 0.020391)
SVM Accuracy: 0.077462 (+/- 0.033696)
Gaussian Process Accuracy: 0.056381 (+/- 0.051491)
Decision Tree Accuracy: 0.049708 (+/- 0.032751)
Random Forest Accuracy: 0.034748 (+/- 0.028303)
Neural Net Accuracy: 0.060194 (+/- 0.033275)
AdaBoost Accuracy: 0.076939 (+/- 0.056166)
Naive Bayes Accuracy: 0.053071 (+/- 0.041586)

### Standardization ###
Nearest Neighbors Accuracy: 0.014245 (+/- 0.017349)
SVM Accuracy: 0.082059 (+/- 0.052744)
Gaussian Process Accuracy: 0.045302 (+/- 0.053427)
Decision Tree Accuracy: 0.047322 (+/- 0.090372)
Random Forest Accuracy: 0.063199 (+/- 0.047135)
Neural Net Accuracy: 0.074308 (+/- 0.064457)
AdaBoost Accuracy: 0.060209 (+/- 0.046526)
Naive Bayes Accuracy: 0.028006 (+/- 0.054382)
```

Hyperparameter Optimization

- Random Search Hyperparameter Optimization
- K-Fold Cross Validation
- Cálculo do resultado

```
clf_model = SVC()

param_dist = {'C': np.random.uniform(low=0.0, high=2.0, size=(10,)),
              'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
              'degree': ss.randint(1, 5),
              'gamma': ['auto', 'scale'],
              'coef0': np.random.uniform(low=0.0, high=2.0, size=(10,)),
              'shrinking': [True, False],
              'probability': [True, False],
              'tol': np.random.uniform(low=0.0, high=2.0, size=(10,)),
              'decision_function_shape': ['ovo', 'ovr']}
```

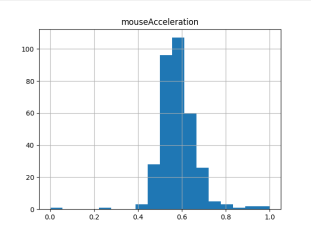
```
rs = RandomizedSearchCV(clf_model, param_distributions=param_dist, n_iter=50, cv=5)
rs.fit(data, target)
report(rs.cv_results_)
```

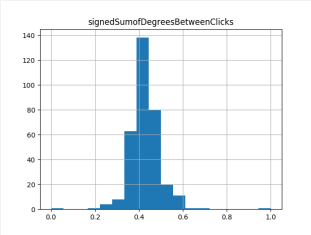
Hyperparameter Optimization

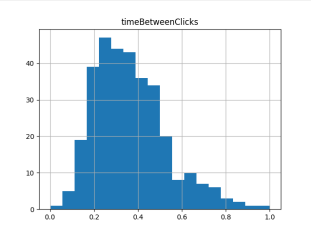
```
Model with rank: 1
Mean validation score: 0.084 (std: 0.033)
Parameters: {'C': 4, 'coef0': 2, 'decision_function_shape': 'ovo', 'degree': 3, 'gamma': 'auto', 'kernel': 'sigmoid', 'probability': True, 'shrinking': True, 'tol': 0.4827764420159888}

Model with rank: 1
Mean validation score: 0.084 (std: 0.006)
Parameters: {'C': 4, 'coef0': 4, 'decision_function_shape': 'ovo', 'degree': 15, 'gamma': 'scale', 'kernel': 'sigmoid', 'probability': False, 'shrinking': True, 'tol': 0.61264535331271}

Model with rank: 3
Mean validation score: 0.081 (std: 0.009)
Parameters: {'C': 1, 'coef0': 9, 'decision_function_shape': 'ovo', 'degree': 10, 'gamma': 'auto', 'kernel': 'sigmoid', 'probability': False, 'shrinking': True, 'tol': 0.3428840777972235}

Model with rank: 3
Mean validation score: 0.081 (std: 0.009)
Parameters: {'C': 7, 'coef0': 7, 'decision_function_shape': 'ovo', 'degree': 2, 'gamma': 'scale', 'kernel': 'sigmoid', 'probability': True, 'shrinking': True, 'tol': 0.15970071338103065}
```

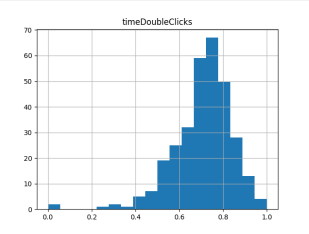
Outras otimizações: feature selection “manual”

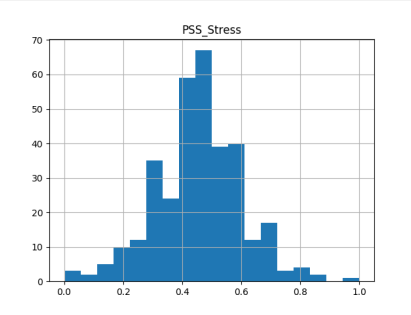
- Escolha das *features* através da observação das distribuições












```
Model with rank: 1
Mean validation score: 0.116 (+/- 0.018)
Parameters: {'C': 0.39834084089582955, 'coef0': 1.4307763023109747, 'decision_function_shape': 'ovo', 'degree': 2, 'gamma': 'auto', 'kernel': 'linear', 'probability': True, 'shrinking': Fals

Model with rank: 1
Mean validation score: 0.116 (+/- 0.018)
Parameters: {'C': 0.39834084089582955, 'coef0': 1.710565575089778, 'decision_function_shape': 'ovo', 'degree': 4, 'gamma': 'auto', 'kernel': 'linear', 'probability': False, 'shrinking': Fals

Model with rank: 3
Mean validation score: 0.104 (+/- 0.043)
Parameters: {'C': 1.7177784155292963, 'coef0': 0.846749364388774, 'decision_function_shape': 'ovo', 'degree': 1, 'gamma': 'auto', 'kernel': 'linear', 'probability': True, 'shrinking': True,

Model with rank: 3
Mean validation score: 0.104 (+/- 0.013)
Parameters: {'C': 0.5672400434577674, 'coef0': 1.4063225404574415, 'decision_function_shape': 'ovr', 'degree': 3, 'gamma': 'scale', 'kernel': 'linear', 'probability': False, 'shrinking': Tru
```

Outras otimizações: binarização do output

- Atribuição de classes à coluna do PSS_Stress:
  - [0–39], “pouco stressado”
  - [40–52], “muito stressado”

```
for i, value in enumerate(target):
    if value > 40:
        target[i] = 1  # muito stressado
    else:
        target[i] = 0  # pouco stressado
```

```
Model with rank: 1
Mean validation score: 0.979 (+/- 0.007)
Parameters: {'C': 1.5079712353671682, 'coef0': 1.7547225296583489, 'decision_function_shape': 'ovr', 'degree': 1, 'gamma': 'auto', 'kernel': 'rbf', 'probability': True, 'shrinking': False,
```

MACHINE LEARNING

Estudo de um dataset sobre dados biométricos de alunos em situação de exame com scikit-learn

André Pereira, Carlos Lemos, João Barreira e Rafael Costa

Aprendizagem e Extração de Conhecimento – Sistemas Inteligentes
2018/2019 – MEI/MIEI – Universidade do Minho