Reconhecimento de Atividades

Wallace Camacho Carlos[1]

[1]Mestrado Profissional em Engenharia de Produção e Sistemas de Computação
Universidade Federal Fluminense (UFF) – Rio de Janeiro, RJ – Brazil

{wallacecamacho}@id.uff.br

Abstract. *Use of the implementation of the recognition of activities using the DataSet (recognition of activity activities of data assembler mounted toast) containing information from accelerometers with columnar dukes of X, Y, Z, movement.*

Resumo. *Utilização da implementação do reconhecimento de atividades utilizando o dataset (Activity Recognition from Single Chest-Mounted Accelerometer Data Set) contendo informações de acelerômetros com dados colunares de movimentos x, y, z, movimento.*

1. Dataset de Reconhecimento de Atividades

Para utilização da implementação do reconhecimento de atividades foi utilizado o dataset (Activity Recognition from Single Chest-Mounted Accelerometer Data Set - https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+from+Single+Chest-Mounted+Accelerometer#) dentro do dataset contém informações do acelerômetros com dados colunares de movimentos x, y, z, movimento.

2. Implementando o Reconhecimento de atividades

Na utilização da implementação foi utilizado a linguagem de programação python com as bibliotecas NumPy, Pandas e SciKitLearn junto com o ambiente de desenvolvimento Google-Colab.

Treinamento dos dados

Os dados foram treinados utilizando os dados do acelerômetro das informações das colunas x, y, z e movimento

```
X = df[['x', 'y', 'z']]
y = df['movimento']
```

Algoritmo – KNN

Na utilização do algoritimo KNN, foi utilizada o parâmetro de n_neighbors = 7, e usando os atributos default na utilização do KNN.

Executando o algoritimo temos o seguinte resultado de acurácia:

Test Accuracy : 0.8352572706935123

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None, n_neighbors=7, p=2,
weights='uniform')

Usando os estimadores hiperparamétricos definimos os seguintes atributos que podem ser utilizados para sua otimização.

leaf_size = list(range(1,20))
n_neighbors = [3, 5, 11]
p=[1,2]#Convert to dictionary
metric = ['euclidian', 'manhattan']
weights = ['uniform', 'distance']

O resultado do algoritmo GridSearch para o KNN, trouxe o resultado dos melhores parâmetros:

Best leaf_size: 3
Best p: 2
Best n_neighbors: 11
Best score: 0.8104161073825503
Best estimator: KNeighborsClassifier(leaf_size=3, n_neighbors=11)
Best params: {'leaf_size': 3, 'n_neighbors': 11, 'weights': 'uniform'}
Test Accuracy : 0.8552125279642058

Algoritmo - Naive Bayes

Utilização do algoritmo com os dados de treinamento X, y; Usando o algoritmo GaussianNB, foram utilizados os parâmetros de cross-validation para validação e controle da acurácia dos dados. Variando a implementação do cv (cross-validation) de 5 até 10 com os respectivos valores de score:

Cross-validation score with 10 folds: [0.66711409 0.82304251 0.83923937 0.84845638 0.81700224]

Mean cross-validation score with 10 folds: 0.799

Cross-validation score with 10 folds: [0.63758221 0.81603508 0.82717364 0.83725883 0.85561263 0.83625658

0.80869456]

Mean cross-validation score with 10 folds: 0.803

Cross-validation score with 10 folds: [0.59128614 0.81034066 0.82813884 0.82008537 0.84239349 0.85672868

0.84552191 0.83835374 0.79663338]

Mean cross-validation score with 10 folds: 0.803

Cross-validation score with 10 folds: [0.58782998 0.79149888 0.83024609 0.81870246 0.83856823 0.84420582

0.85959732 0.83847875 0.84178971 0.79069351]

Mean cross-validation score with 10 folds: 0.804

Resultado do teste de acurácia:

Test Accuracy : 0.8103205162969792

Algoritmo – RandonForest

Na implementação do algoritimo foram utilizados os critérios de entropia e gini,

Critério gini, min_samples_split=5, min_samples_leaf=2 , max_depth=None.

Acurácia sobre o test:

Test Accuracy : 0.8086835134941657

Critério entropia, min_samples_split=5, min_samples_leaf=2 , max_depth=None.

Acurácia sobre o test:

Test Accuracy : 0.8062137590378695

Utilização do algoritmo GridSearch para hiper-parametrização

'max_depth': [2, 3, 5, 10, 20],

'min_samples_leaf': [5, 10, 20, 50, 100],

'criterion': ["gini", "entropy"]

DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
max_depth=20, max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=20, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort='deprecated',
random_state=42, splitter='best')

Test Accuracy : 0.8573985253060348

2.1. Resultados do Teste de Acurácia

Tabela 1 – Comparativo Acurácia

Algoritmo	Acurácia
RandomForest	0.8573985253060348
KNN	*0.8552125279642058*
GaussianNB	0.8103205162969792

O algoritmo mostrando um melhor resultado de acurário utilizando os hiper-parâmetros com total de 0.8573985253060348.

Referências

GOMES, Eduardo et al. Machine Learning Algorithms for Activity-Intensity Recognition Using Accelerometer Data. **MDPI**, 09/02 2021. Acesso em: 4 set. 2021.