Klasyfikacja aktywności fizycznej

L Metody i narzędzia Big Data, wt 15:15
Jędrzej Jamnicki | 254290

1. Opis projektu

Celem projektu było porównanie efektywności klasyfikacji danych w zależności od użytej metody redukcji wymiarów zbioru danych.

2. Narzędzia

Do implementacji użyto języka programowania Python. Skorzystano z bibliotek:

- pandas
- NumPy
- SciPy
- matplotlib
- scikit-learn

3. Badany zbiór danych

Dane zostały zebrane podczas eksperymentu przeprowadzonym na grupie 30 ochotników w przedziale wiekowym 19-48 lat. Każda z osób wykonywała sześć czynności mając na pasie smartfon wyposażony w czujniki. Zbiór danych reprezentują wektory odczytów z akcelerometru i żyroskopu oraz ich podstawowe statystyki. Dane zostały uprzednio znormalizowane oraz ustandaryzowane. Zbiór danych został losowo podzielony na dwa zestawy, gdzie 70% ochotników zostało wybranych do wygenerowania danych treningowych, a 30% danych testowych. Jedna linijka pliku tekstowego odpowiada pojedynczemu wektorowi danych.

Źródło: *link*

4. Przebieg implementacji

Pierwszym krokiem była ekstrakcja cech i etykiet z plików tekstowych oraz zapisanie danych w pamięci.

```
features = np.array(get_features(FEATURES_FILE_PATH))

X_train = np.array(extract_lines_val(X_TRAIN_FILE_PATH))
y_train = np.array(extract_labels(Y_TRAIN_FILE_PATH))

X_test = np.array(extract_lines_val(X_TEST_FILE_PATH))
y_test = np.array(extract_labels(Y_TEST_FILE_PATH))
```

Należało sprawdzić również czy dane są kompletne i nie wymagają uzupełnienia.

```
any(train_df.isnull().any()), any(test_df.isnull().any())
```

```
(False, False)
```

Dane zostały uprzednio znormalizowane oraz ustandaryzowane więc były gotowe do redukcji wymiarowości za pomocą metod analizy głównych składowych (PCA) oraz liniowej analizy dyskryminacyjnej (LDA).

```
pca = PCA(n_components=2)
principal_components = pca.fit_transform(X_train)
pca_df = pd.DataFrame(data=principal_components,
                      columns=['Principal component 1', 'Principal component 2'])
pca_df['Activity'] = y_train
```

```
LDA_model = LDA(n_components=2)
lda_components = LDA_model.fit_transform(X_train, y_train)
lda_df = pd.DataFrame(data=lda_components,
                      columns=['Principal component 1', 'Principal component 2'])
lda_df['Activity'] = y_train
```

Następnie przystąpiono do klasyfikacji za pomocą algorytmów LDA i k-najbliższych sąsiadów (KNN) dla danych przed redukcją wymiarowości oraz po redukcji.

Implementacja algorytmu KNN:

```
class KNN():
    def __init__(self, N=5):
        self.N = N

    def fit(self, X_train, Y_train):
        self.X_train = X_train
        self.Y_train = Y_train
        self.m, self.n = X_train.shape

    def predict(self, X_test):
        self.X_test = X_test
        self.m_test, self.n = X_test.shape

        Y_predict = np.zeros(self.m_test)
        for i in range(self.m_test):
            x = self.X_test[i]
            neighbors = np.zeros(self.N)
            neighbors = self.find_neighbors(x)
            Y_predict[i] = mode(neighbors)[0][0]

        return Y_predict

    def find_neighbors(self, x):
        euclidean_distances = np.zeros(self.m)
        for i in range(self.m):
            d = self.euclidean(x, self.X_train[i])
            euclidean_distances[i] = d

        inds = euclidean_distances.argsort()
        Y_train_sorted = self.Y_train[inds]

        return Y_train_sorted[:self.N]

    def euclidean(self, x, x_train):
        return np.sqrt(np.sum(np.square(x-x_train)))
```

Niezmodyfikowane dane treningowe:

```
lda_predicted = LDA_model.predict(X_test)
```

```
my_knn = KNN()
my_knn.fit(X_train, y_train)
my_knn_pred = my_knn.predict(X_test)
```

Dane treningowe zredukowane przez algorytm PCA:

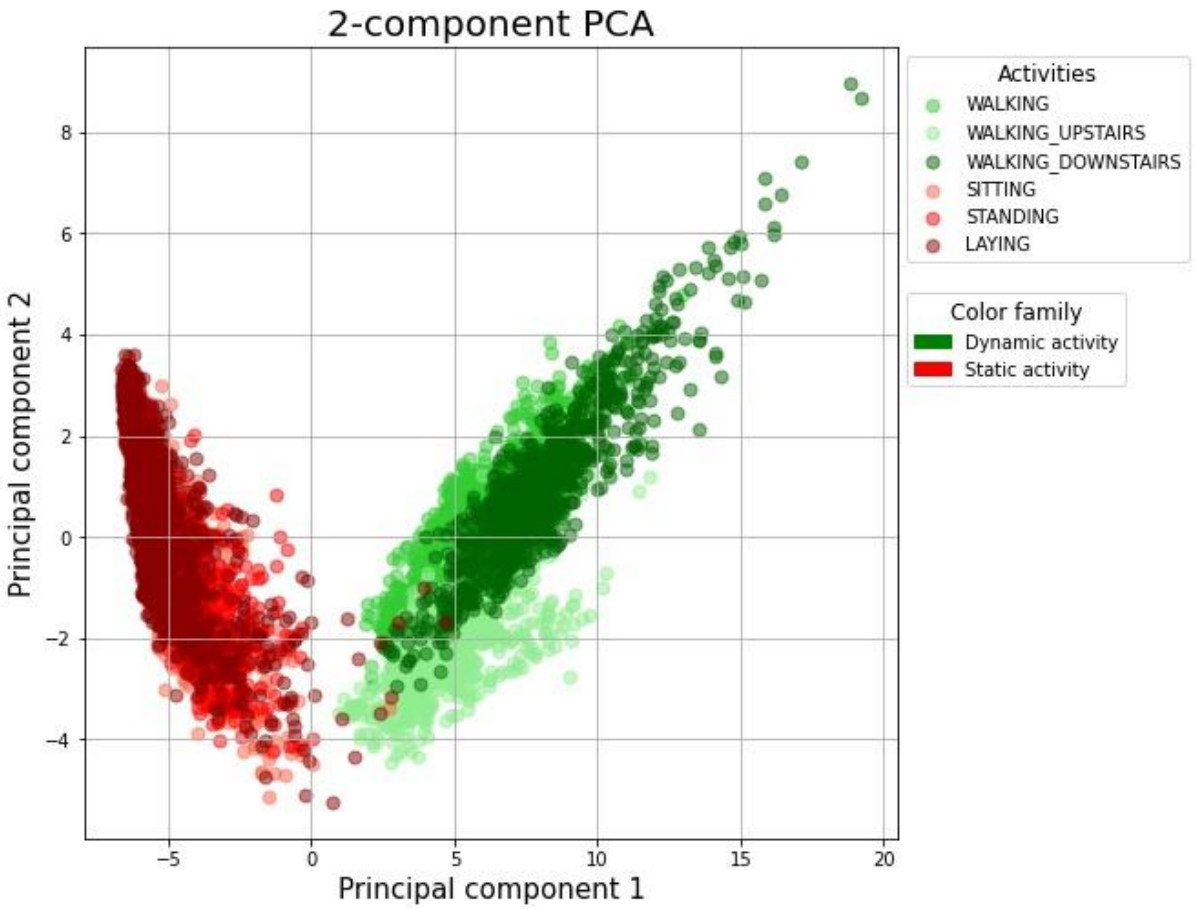
```
my_knn.fit(np.array(pca_df[['Principal component 1', 'Principal component 2']]), y_train)
my_knn_pred_pca = my_knn.predict(pca.transform(X_test))
```

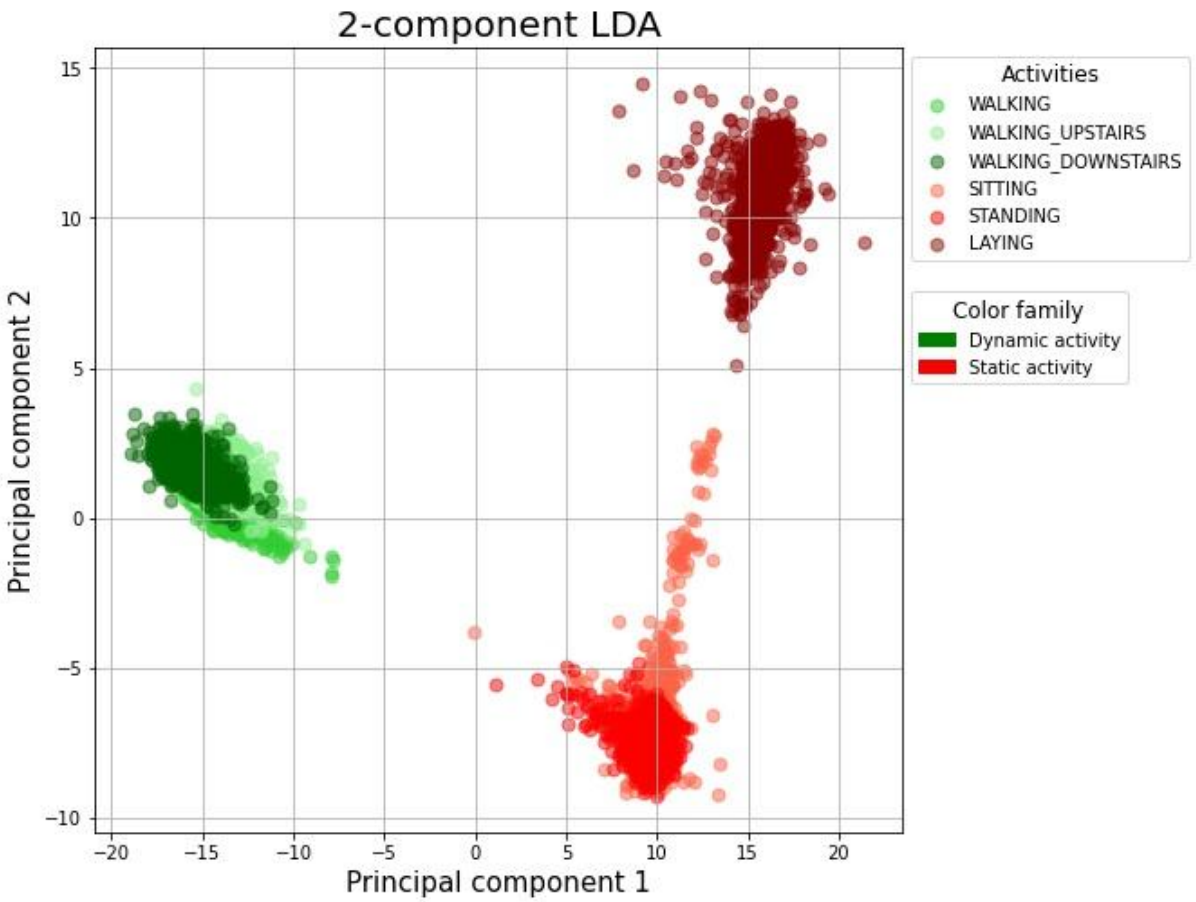
Dane treningowe zredukowane przez algorytm LDA:

```
my_knn.fit(np.array(lda_df[['Principal component 1', 'Principal component 2']]), y_train)
my_knn_pred_lda = my_knn.predict(LDA_model.transform(X_test))
```

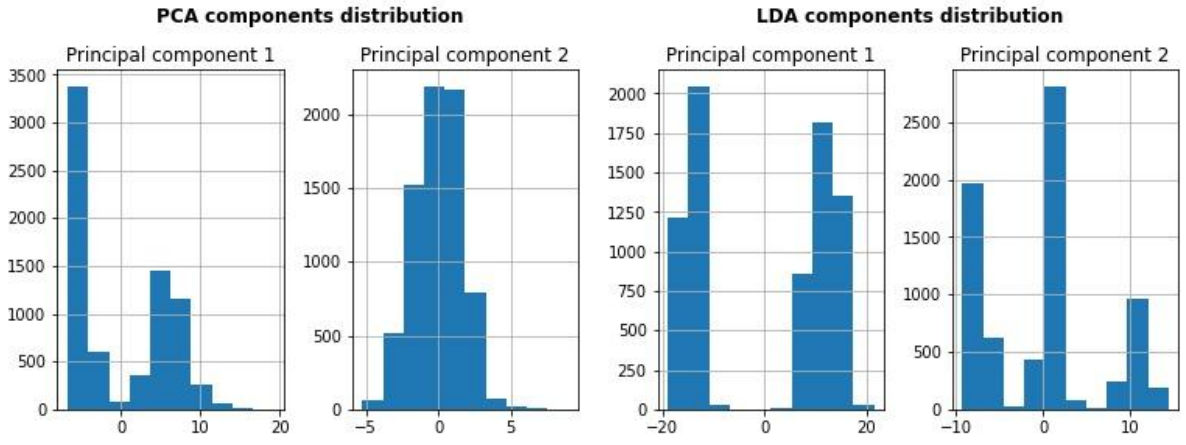
5. Przedstawienie wyników

Wykresy głównych składowych po redukcji wymiarowości:





Rozkłady głównych składowych:



Wyniki klasyfikacji:

	WALKING	WALKING_UPSTAIRS	WALKING_DOWNSTAIRS	SITTING	STANDING	LAYING	OVERALL
LDA	0.9939	0.9891	0.9949	0.9732	0.9735	1	0.9874
KNN	0.9674	0.9671	0.9640	0.9522	0.9535	0.9990	0.9672
KNN+PCA	0.8738	0.9382	0.8789	0.7577	0.7950	0.8537	0.8496
KNN+LDA	0.8249	0.8378	0.8897	0.8341	0.8344	1	0.8702

6. Analiza wyników

W przypadku selekcji cech głównych przy użyciu PCA można zauważyć wyraźną odwrotność zależności między dwoma typami aktywności (statycznym i dynamicznym) co wydaje się być zgodne z logiką. Mniej wyraźnie ale również w przypadku LDA można zauważyć podział na aktywności statyczne i dynamiczne. Klasyfikatorem z największą dokładnością okazał się algorytm LDA, zaś z najniższą KNN po uprzednim zredukowaniu wymiarowości danych treningowych przez PCA.

7. Wnioski

Metody redukcji wymiarowości danych są w stanie w znaczącym stopniu skrócić czas klasyfikacji. Metody te są użyteczne w szczególności podczas pracy na ogromnych zbiorach danych. Trzeba jednak liczyć się z utratą informacji a więc także z obniżoną dokładnością klasyfikatora.