AutoML projekt 2

Jakub Kępka, Damian Kąkol

Grupa docelowa

- Pakiet AutoML jest skierowany dla problemu multilabel classifcation dla danych sekwencyjnych.
- Pakiet umożliwia wypłaszczanie danych w postaci surowej o kształcie (sekwencje, próbki, cechy)

Definicja problemu

- Nasz model ma za zadanie podzielić szereg na stałego rozmiaru (rozmiar można dowolnie wybierać) okna, następnie na podstawie danych z okna dokonujemy klasyfikacji.
- Input: (n_frames, n_features)
- Output: (n_windows_size, n_classes)
  - Macierz 0 i 1

Problematyka

- Celem pakietu jest dostowanie problemu klasyfikacji wieloeytkietowej na danych o silnych zależnościach czasowych do problemu, który może być rozwiązany typowem pakietem automl, dodatkowo chcemy przygotować modele skorojone pod ten problem.

Podobne pakiety

- https://automl.github.io/auto-sklearn/master/examples/20_basic/example_multilabel_classification.html - auto sklearn udostępnia klase AutoSklearnClassifier, która według dokumentacji działa samoistnie na danych wieloetykietowych, niemniej dla naszych danych nie poradziła sobie naj
- Nie wydaje się żeby istniał projekt open-source dla danych sekwencyjnych

```
# Using reuters multilabel dataset -- https://www.openml.org/d/40594
X, y = sklearn.datasets.fetch_openml(data_id=40594, return_X_y=True, as_frame=False)

# fetch openml downloads a numpy array with TRUE/FALSE strings. Re-map it to
# integer dtype with ones and zeros
# This is to comply with Scikit-learn requirement:
# "Positive classes are indicated with 1 and negative classes with 0 or -1."
# More information on: https://scikit-learn.org/stable/modules/multiclass.html
y[y == "TRUE"] = 1
y[y == "FALSE"] = 0
y = y.astype(int)

# Using type of target is a good way to make sure your data
# is properly formatted
print(f"type_of_target={type_of_target(y)}")

X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(
    X, y, random_state=1
)
```

```
Out: type_of_target=multilabel-indicator
```

Building the classifier

```
automl = autosklearn.classification.AutoSklearnClassifier(
    time_left_for_this_task=60,
    per_run_time_limit=30,
    # Bellow two flags are provided to speed up calculations
    # Not recommended for a real implementation
    initial_configurations_via_metalearning=0,
    smac_scenario_args={"runcount_limit": 1},
)
```

Istniejące rozwiązania

- **Human Action Evaluation applied to Weightlifting**
- Projekt skupiał się na klasyfikowaniu błedów w wykonywaniu pewnych ćwiczeń siłowych na podstawie analizy nagrania.
- Dane wejściowe dla tego problemu stanowi macierz rozmiaru (n_frames, n_features).
- Nagranie dzielone zostaje na określonej długości okna, okna następnie są klasyfikowane.

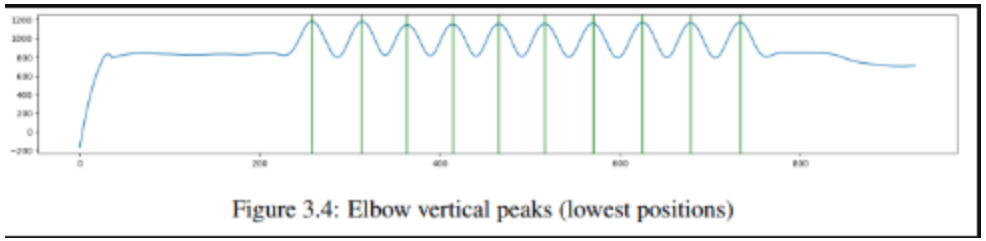


Figure 3.4: Elbow vertical peaks (lowest positions)

Istniejące rozwiązania

- **HAR - Human Activity Recognition**
- Dane pochodzące z akcelerometrów i żyroskopów (np. w telefonach komórkowych lub opaskach fitness).
- Typowy format danych to macierz (n_frames, n_features), gdzie n_features to liczba kanałów (np. akcelerometr x, y, z). Tworzą zatem dla odpowiednie wejście.
- Dzieli się dane na okna (np. co 2 sekundy).
- Każde okno jest klasyfikowane jako jedna z aktywności (np. chodzenie, bieganie, siedzenie).

Istniejące biblioteki

- **tsai**

- Tsai to open-source'owy pakiet do głębokiego uczenia, zbudowany na bazie Pytorch i fastai, skupiający się na najnowszych technikach stosowanych w zadaniach związanych z szeregami czasowymi, takich jak klasyfikacja, regresja, prognozowanie.

- **FastAI**

- popularny framework do głębokiego uczenia, który oferuje prostotę użycia, jednocześnie dając użytkownikowi dużą elastyczność. FastAI jest zbudowane na PyTorch, co daje nam możliwość tworzenia modeli deep learningowych.

Istniejące biblioteki

- **Random Forest Classifier dla szeregów czasowych**
- Random forest to estymator, który dopasowuje szereg klasyfikatorów drzew decyzyjnych do różnych podpróbek zbioru danych i wykorzystuje średnią, aby poprawić dokładność predykcyjną oraz kontrolować nadmierne dopasowanie (overfitting).
- Ten transformator wyodrębnia 3 cechy z każdego okna: średnią, odchylenie standardowe i nachylenie. Całkowita liczba cech wynosi więc 3 * liczba_okien. Następnie budowane jest las losowy, wykorzystujący te cechy jako dane wejściowe.

Narzędzia

- **MultiOutputClassifier – autosklearn**
- Biblioteka Auto-SKlearn udostępnie możliwość z korzystania z klasy MultiOutputClassifier która to pozwala na używania jednego klasyfikatora do jednego celu.
- Funkcja fit tego klasyfikatora pasuje dla docelowego kształtu naszych danych - zarówno wejścia i wyjścia.
- **OneVsRestClassifier – scikitlearn**
- **Iterative_train_test_split - scikit-multilearn**
  ta funkcja zapewnia strafytikacje danych wieloetykietowych przy podziale na zbiory testowe i treningowe, zachowuje ona rozkład w okolicach błędu 1 punktu procentowego

Przetwarzanie danych

- Główna klasa **AutoMlMultiLabelClassifier** udostępnia metodę, która wypłaszcza dane do postaci dwuwymiarowej. Rozmiar okna do wypłaszczania jest sterowalny za pomocą argumentu fraction. Z powodu ograniczeń technicznych ta funkcjonalność nie może być częścią pipline'u
- W pipelinach dane są redukowane do 80 głównych komponentów za pomocą PCA, chyba że cech jest jest mniej, wtedy pozostają orginalne cechy.
- Dodatkowo uzupełniane są brakujące wartości , oraz dane podlegają standaryzacji

```
feature_pipeline = Pipeline([
                ('imputer', SimpleImputer(strategy='mean')),
                ('scaler', StandardScaler()),
                ('pca', PCADimensionReducer()),
            ])
```

Selekcja i optymalizacja modeli

Modele przygotowane dla problemu to:

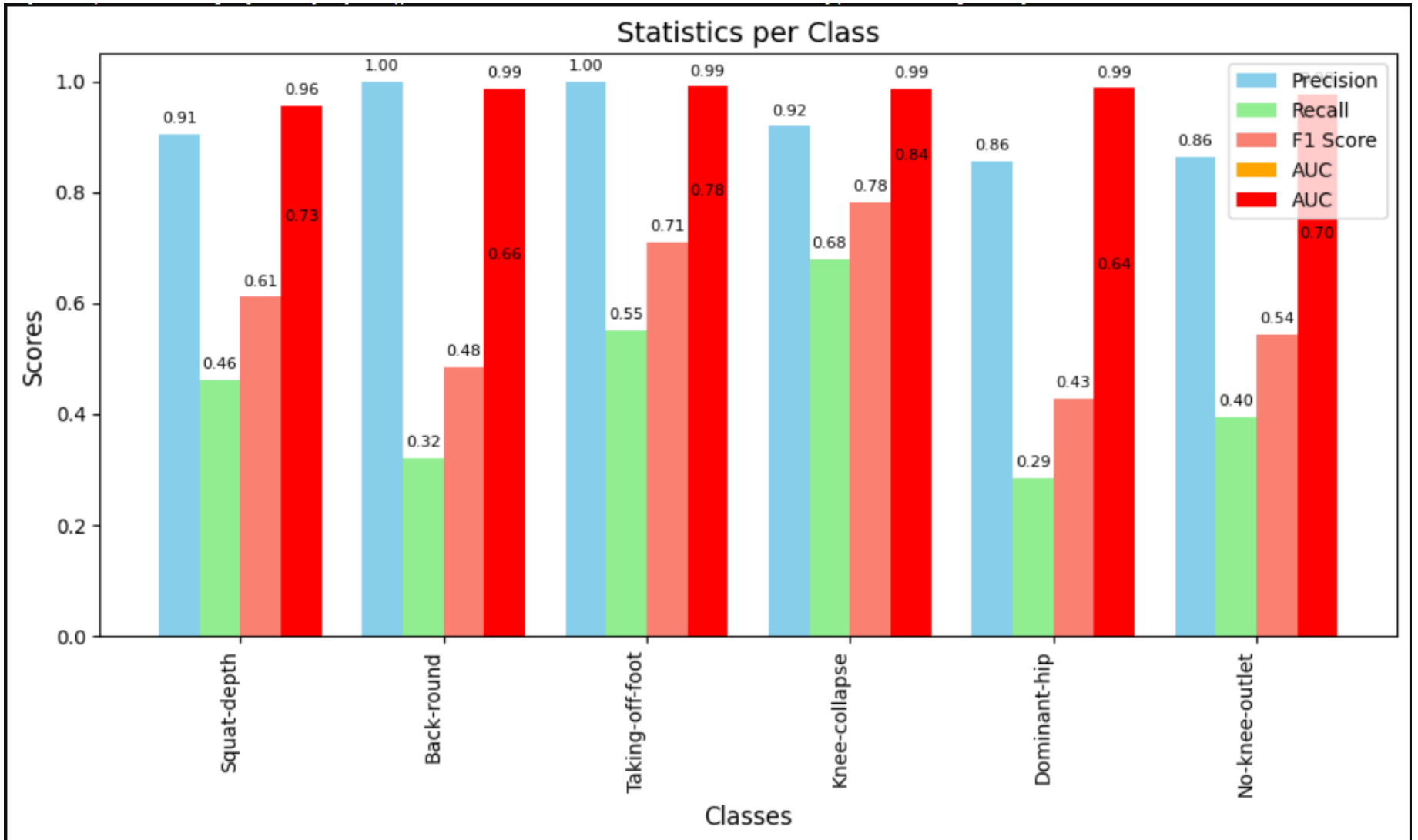
- SVC OneVsRest
- Regresja logistyczna OneVsRest
- XGBoost OneVsRest
- XGBoost Mulitoutput

Optymalizacja odbywa się za pomocą grid search i znanej siatki hiperparametrów

Raport wyników najlepszego modelu

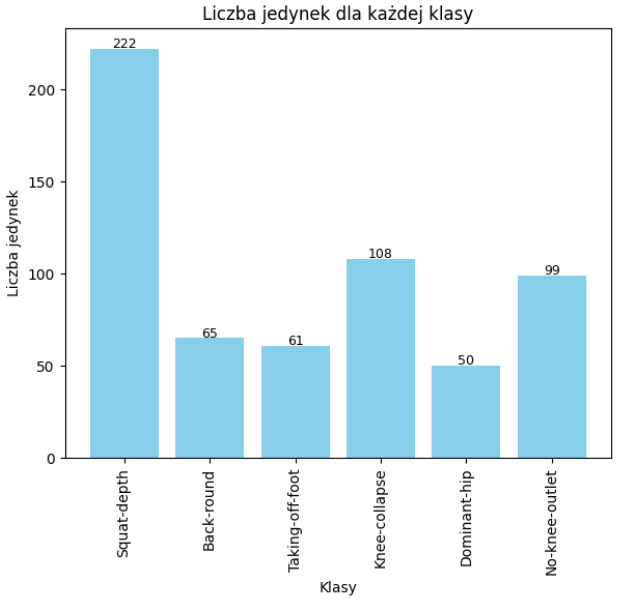
```
--------------------------------------------------

Models ranking:
1. SVC - Best CV Score: 0.6674703873341373 - Best Parameters: {'model__estimator__C': 100, 'model__estimator__gamma': 'scale', 'model__estimator__kernel': 'rbf'}
2. OneVsRest_LogisticRegression - Best CV Score: 0.4999607716425934 - Best Parameters: {'model__estimator__C': 10, 'model__estimator__max_iter': 10000, 'model__estimator__penalty':
'l2', 'model__estimator__solver': 'saga'}
3. XGBoost_MultiOutput - Best CV Score: 0.42429977325016577 - Best Parameters: {'model__n_estimators': 1500}
4. XGBoost - Best CV Score: 0.3392059954206418 - Best Parameters: {'model__estimator__n_estimators': 500}
---------------------------------------
Best Model: OneVsRestClassifier
Model Parameters:
  estimator__C: 100
  estimator__break_ties: False
  estimator__cache_size: 200
  estimator__class_weight: None
  estimator__coef0: 0.0
  estimator__decision_function_shape: ovr
  estimator__degree: 3
  estimator__gamma: scale
  estimator__kernel: rbf
  estimator__max_iter: -1
  estimator__probability: True
  estimator__random_state: None
  estimator__shrinking: True
  estimator__tol: 0.001
  estimator__verbose: False
  estimator: SVC(C=100, probability=True)
  n_jobs: None
  verbose: 0
---------------------------------------
Test Set statistics
Accuracy: 0.9041095890410958
Recall: 0.6630434782608695
Precision: 0.7774665551839466
```

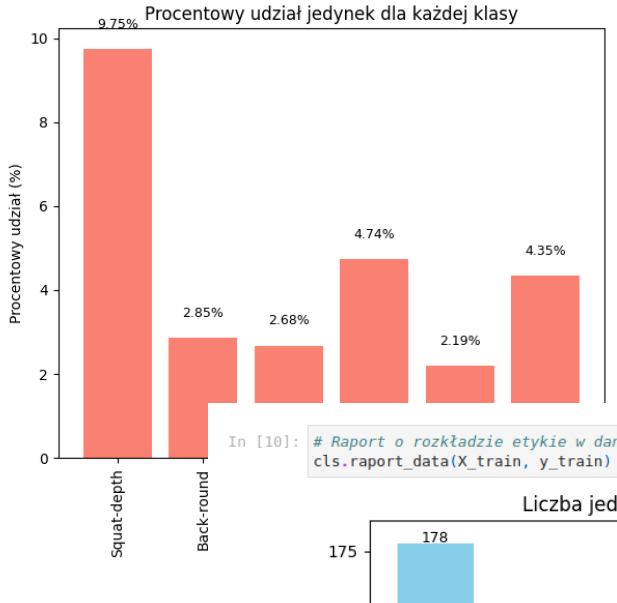
Raport wyników najlepszego modelu



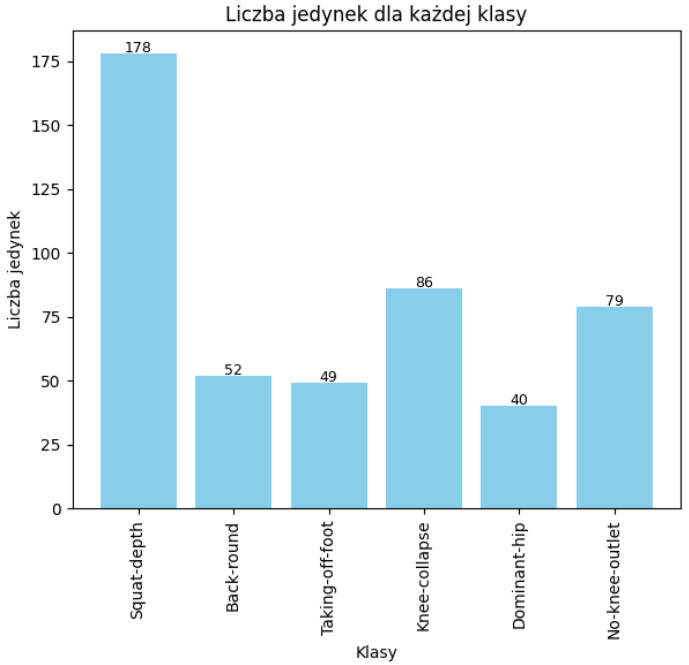
Raport o rozkładzie klas w danych

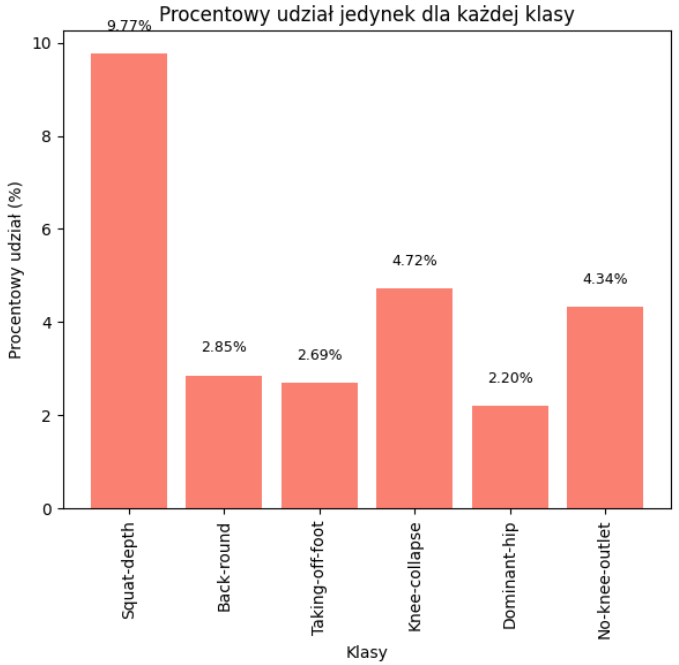
```
[9]: # Raport o rozkładzie etykie w danych
cls.raport_data(X, y)
```






```
In [10]: # Raport o rozkładzie etykie w danych
cls.raport_data(X_train, y_train)
```





Koniec