	Politechnika Bydgoska im. J.J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki		
Przedmiot	Algorytmy i eksploracja danych		
Prowadzący	dr inż. Michał Kruczkowski		
Temat	Uczenie zespołowe		
Studenci	Adam Szreiber, Cezary Naskręt		
Nr ćw.	5	**Data wykonania**	21.10.2023

1. Cel ćwiczenia

Uczenie zespołowe to zaawansowany obszar uczenia maszynowego, w którym wiele modeli jest łączonych w celu poprawy jakości predykcji lub rozwiązania bardziej złożonych problemów.
Laboratorium miało na celu zapoznanie się z podstawowymi metodami uczenia zespołowego. W trakcie laboratorium mieliśmy okazję zapoznać się z różnymi technikami takimi jak Bagging, Random Forest, Boosting, czy Gradient Boosting. Jednym z głównych punktów programu była metoda VotingClassifier, która pozwala na agregację wyników kilku klasyfikatorów w celu uzyskania bardziej stabilnych i dokładnych prognoz.

2. Przebieg laboratorium

2.1. Zadanie 1

Zbiór danych:
https://archive.ics.uci.edu/dataset/341/smartphone+based+recognition+of+human+activities+and+postural+transitions

- Treningowy: X_train.txt / y_train.txt
- Testowy: X_test.txt / y_test.txt

2.2. Zadanie 2

Wykorzystując zbiór danych z poprzednich laboratoriów wykonaj uczenie zespołowe dla w następującej konfiguracji:

1. Liczba klasyfikatorów: 3
2. Algorytmy klasyfikacji: dowolne
3. Trenowanie: każdy klasyfikator na całym zbiorze treningowym

W ramach uczenia zespołowego wypisz wypadkową ocenę stosując zasadę większości głosów.

Jako rezultat przedstaw wynik kroswalidacji dla 5 podprób oraz wynik testowania na zbiorze testowym.

Jako kryterium oceny klasyfikacji wypisz następujące parametry:
• ACC
• Recall
• F1
• AUC

Na początku należało zaimportować wymagane biblioteki i funkcje

```
from sklearn.metrics import accuracy_score
import numpy as np
from sklearn.ensemble import VotingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, recall_score, f1_score, roc_auc_score
import pandas as pd


# Wczytaj dane treningowe i testowe
X_train = np.loadtxt('X_train.txt')
y_train = np.loadtxt('y_train.txt')
X_test = np.loadtxt('X_test.txt')
y_test = np.loadtxt('y_test.txt')
```

Stworzenie klasyfikatorów i wytrenowanie danymi.

```
# Stwórz klasyfikatory
bagging_classifier = BaggingClassifier(
    estimator=GaussianNB(), n_estimators=3)
random_forest_classifier = RandomForestClassifier(n_estimators=3)
gb_classifier = GradientBoostingClassifier(max_depth=1, n_estimators=5)

# Trenuj każdy klasyfikator na całym zbiorze treningowym
bagging_classifier.fit(X_train, y_train)
random_forest_classifier.fit(X_train, y_train)
gb_classifier.fit(X_train, y_train)
```

Weryfikacja wytrenowanych modeli na danych testowych

```
# Przeprowadź prognozowanie na zbiorze testowym
y_pred_bagging = bagging_classifier.predict(X_test)
y_pred_random_forest = random_forest_classifier.predict(X_test)
y_pred_bayes_optimal = gb_classifier.predict(X_test)
```

Wykonanie głosowania większościowego i wyświetlenie dokładności

```
# Możesz wykonać głosowanie większościowe (np. dla klasyfikacji binarnej)
y_ensemble = (y_pred_bagging + y_pred_random_forest +
              y_pred_bayes_optimal >= 2).astype(int)

# Oblicz dokładność
ensemble_accuracy = accuracy_score(y_test, y_ensemble)
print("Dokładność modelu zespołowego: {:.2f}".format(ensemble_accuracy))
```

Dodanie kodu kroswalidacji danych i wyliczenie oceny większościowej

```
# Przeprowadź kroswalidację dla 5 podprób
y_pred_bagging_cv = cross_val_predict(
    bagging_classifier, X_train, y_train, cv=5)
y_pred_random_forest_cv = cross_val_predict(
    random_forest_classifier, X_train, y_train, cv=5)
y_pred_bayes_optimal_cv = cross_val_predict(
    gb_classifier, X_train, y_train, cv=5)

# Oblicz wypadkową ocenę stosując zasadę większości głosów na podstawie wyników kroswalidacji
ensemble_result_cv = (y_pred_bagging_cv + y_pred_random_forest_cv +
                      y_pred_bayes_optimal_cv >= 2).astype(int)
```

W kolejnym kroku liczymy i wyświetlamy kryteria oceny takie jak: recall score, accuracy score, f1 score.

```
# Oblicz dokładność wypadkowej oceny w kroswalidacji
ensemble_accuracy_cv = accuracy_score(y_train, ensemble_result_cv)

# Oblicz czułość (Recall) w kroswalidacji
ensemble_recall_cv = recall_score(
    y_train, ensemble_result_cv, average='weighted')

# Oblicz F1 Score w kroswalidacji
ensemble_f1_score_cv = f1_score(
    y_train, ensemble_result_cv, average='weighted')

print("Dokładność modelu zespołowego w kroswalidacji (ACC): {:.2f}".format(
    ensemble_accuracy_cv))
print("Czułość modelu zespołowego w kroswalidacji (Recall): {:.2f}".format(
    ensemble_recall_cv))
print("F1 Score modelu zespołowego w kroswalidacji: {:.2f}".format(
    ensemble_f1_score_cv))
```

Należało również przeprowadzić predykcję na zbiorze testowym i ocenić stosując większość głosów co uczyniono poniżej.

```
# Przeprowadź prognozowanie na zbiorze testowym
y_pred_bagging_test = bagging_classifier.predict(X_test)
y_pred_random_forest_test = random_forest_classifier.predict(X_test)
y_pred_bayes_optimal_test = gb_classifier.predict(X_test)

# Oblicz wypadkową ocenę stosując zasadę większości głosów na podstawie wyników na zb
ensemble_result_test = (y_pred_bagging_test + y_pred_random_forest_test +
                        y_pred_bayes_optimal_test >= 2).astype(int)
```

Ostatnim krokiem była ocena wyuczonego modelu co przedstawiono poniżej

```
# Oblicz dokładność wypadkowej oceny na zbiorze testowym
ensemble_accuracy_test = accuracy_score(y_test, ensemble_result_test)

# Oblicz czułość (Recall) na zbiorze testowym
ensemble_recall_test = recall_score(
    y_test, ensemble_result_test, average='weighted')

# Oblicz F1 Score na zbiorze testowym
ensemble_f1_score_test = f1_score(
    y_test, ensemble_result_test, average='weighted')


votingClass = VotingClassifier(estimators=[
    ('bagging', bagging_classifier),
    ('random_forest', random_forest_classifier),
    ('gb', gb_classifier)
], voting='hard')  # Ustawienie 'hard' oznacza zasadę większości głosów

votingClass.fit(X_train,y_train)

# # Oblicz AUC (Area Under the ROC Curve) na zbiorze testowym
ensemble_auc_test = roc_auc_score(
    y_test, votingClass.predict_proba(X_test), multi_class='ovr')
```

Tworzenie obiektu danych w który zostanie zapisany do pliku

```
# Tworzenie DataFrame
data = {
    'Metryka': ['Accuracy', 'Recall', 'F1 Score', 'AUC'],
    'Kroswalidacja': [ensemble_accuracy_cv, ensemble_recall_cv, ensemble_f1_score_cv, 'error'],
    'Testowanie': [ensemble_accuracy_test, ensemble_recall_test, ensemble_f1_score_test, 'error']
}

df = pd.DataFrame(data)

# Zapis do pliku CSV (CSV)
df.to_excel('ensemble_learning.xlsx', index=False)
```

Wyniki wykonywania programu

	A	B	C
1	Metryka	Kroswalidacja	Testowanie
2	Accuracy	0.16	0.16
3	Recall	0.16	0.16
4	F1 Score	0.04	0.04
5	AUC	0.98	0.99

2.3. Zadanie 3

Wykorzystując zbiór danych z poprzednich laboratoriów wykonaj uczenie zespołowe stosując agregację klasyfikatorów dla w następującej konfiguracji:
4. Liczba klasyfikatorów: 3
5. Algorytmy klasyfikacji: dowolne
6. Trenowanie: na losowo wygenerowanych podzbiorach

W ramach agregacji klasyfikatorów wypisz wypadkową ocenę stosując zasadę większości głosów.
Jako rezultat przedstaw wynik kroswalidacji dla 5 podprób oraz wynik testowania na zbiorze testowym.
Jako kryterium oceny klasyfikacji wypisz następujące parametry:

- ACC
- Recall
- F1
- AUC

W tym zadaniu pominąłem ładowanie plików ponieważ nic się nie zmieniło względem poprzedniego zadania.
Poniżej przedstawiono przygotowanie klasyfikatorów i utworzenie klasyfikatora głosowania większościowego zawierającego w sobie następujące klasyfikatory: Random Forest, AdaBoost, Gradient Boosting, Bagging

```
# Definicja klasyfikatorów
rf_clf = RandomForestClassifier(random_state=42, n_estimators=3,max_depth=1)
ada_clf = AdaBoostClassifier(random_state=42, n_estimators=3)
gb_clf = GradientBoostingClassifier(random_state=42, n_estimators=3, max_depth=1)
bagging_clf = BaggingClassifier(estimator=rf_clf, n_estimators=3, random_state=4)

# Konfiguracja głosowania
ensemble_clf = VotingClassifier(estimators=[
    ('Random Forest', rf_clf),
    ('AdaBoost', ada_clf),
    ('Gradient Boosting', gb_clf),
    ('Bagging', bagging_clf)
], voting='hard')  # 'hard' oznacza głosowanie większościowe
```

Po utworzeniu klasyfikatorów następnym krokiem było wykonanie trenowania i przewidywania.

```python
# Trenowanie modelu ensemble
ensemble_clf.fit(X_train, y_train)

# Przewidywanie na zbiorze testowym
y_pred = ensemble_clf.predict(X_test)
```

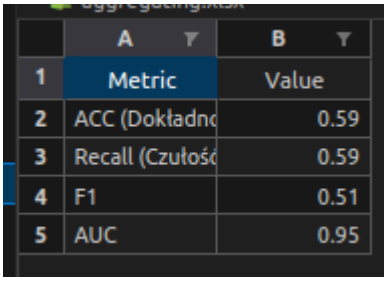
W końcowej części kodu dodano obliczanie oceny klasyfikacji dla metod: accuracy, recall, f1.

```python
# Obliczanie dokładności
accuracy = accuracy_score(y_test, y_pred)
print("Dokładność modelu ensemble: ", accuracy)

# Obliczanie Recall (czułość)
recall = recall_score(y_test, y_pred, average='weighted')
print("Recall (Czułość):", recall)

# Obliczanie F1
f1 = f1_score(y_test, y_pred, average='weighted')
print("F1:", f1)

# Obliczanie AUC
auc = roc_auc_score(y_test, ensemble_clf.predict_proba(X_test), average='weighted', multi_class='ovr')
print("AUC:", recall)

# Wynik kroswalidacji
cross_val_scores = cross_val_score(ensemble_clf, X_train, y_train, cv=5)
print("Wyniki kroswalidacji (5 podprób):", cross_val_scores)
```

Dodanie kodu który wygeneruje plik xlsx i zapisze wyniki.

```python
# Tworzenie ramki danych z wynikami
results = pd.DataFrame({
    'Metric': ['ACC (Dokładność)', 'Recall (Czułość)', 'F1'],
    'Value': [accuracy, recall, f1]
})

# Zapisanie wyników do pliku Excel (xlsx)
results.to_excel('aggregating.xlsx', index=False)
```

Poniżej przedstawienie danych zapisanych do pliku excel



	A	B
1	Metric	Value
2	ACC (Dokładnc	0.59
3	Recall (Czułość	0.59
4	F1	0.51
5	AUC	0.95

2.4. Zadanie 4

Wykonaj tożsame zadanie klasyfikacji jak w poprzednich ćwiczeniach dla następujących metod uczenia zespołowego przyjmując domyślne parametry:

- ADABoost
- XGBoost

Rezultaty zapisz do pliku o nazwie boosting z rozszerzeniem xls lub xlsx

Tak jak w poprzednim zadaniu zostanie pominięte wczytywanie pliku.
Poniżej kod realizujący trenowanie, testowanie i sprawdzenie dokładności modelu ADABoost.

```
# Inicjalizacja modelu ADABoost z domyślnymi parametrami
ada_boost_model = AdaBoostClassifier()

# Trenowanie modelu na danych treningowych
ada_boost_model.fit(X_train, y_train)

# Prognozowanie na danych testowych
y_pred = ada_boost_model.predict(X_test)

# Oblicz dokładność modelu
accuracy = accuracy_score(y_test, y_pred)
print(f'Dokładność modelu ADABoost (z domyślnymi parametrami): {accuracy}')
```

Na kolejnym screenie został ukazany kod realizujący inicjalizację, uczenie, prognozowanie, i sprawdzanie dokładności wytrenowania przy pomocy klasyfikatora XGB.

```
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.transform(y_test)

# Inicjalizacja modelu XGBoost z domyślnymi parametrami
xgb_model = xgb.XGBClassifier()

# Trenowanie modelu na danych treningowych
xgb_model.fit(X_train, y_train)

# Prognozowanie na danych testowych
y_pred = xgb_model.predict(X_test)

# Oblicz dokładność modelu
accuracy2 = accuracy_score(y_test, y_pred)
print(f'Dokładność modelu XGBoost (z domyślnymi parametrami): {accuracy2}')
```

Ostatnim krokiem jest zapisanie wyników do pliku, co wygląda następująco

```
# Tworzenie ramki danych z wynikami
results = pd.DataFrame({
    'Metric': ['AdaBoosting Accuracy', 'XGBoost accuracy'],
    'Value': [accuracy1, accuracy2]
})

# Zapisanie wyników do pliku Excel (xlsx)
results.to_excel('boosting.xlsx', index=False)
```

2.5. Zadanie 5

Porównaj wyniki klasyfikacji wszystkich metod uczenia zespołowego w zbiorczej tabeli na podstawie następujących kryteriów:

- ACC
- Recall
- F1
- AUC Rezultaty zapisz do pliku o nazwie comparison z rozszerzeniem xls lub xlsx

Po raz kolejny pominięto fazę wczytywania danych.

Poniżej kod inicjalizujący modele wszystkie modele uczenia

```
# Przygotuj modele
rf_clf = RandomForestClassifier(random_state=42, n_estimators=20)
ada_clf = AdaBoostClassifier(random_state=42, n_estimators=20)
gb_clf = GradientBoostingClassifier(random_state=42, n_estimators=20)
bagging_clf = BaggingClassifier(base_estimator=rf_clf, n_estimators=20, random_state=42)
extra_trees_clf = ExtraTreesClassifier(random_state=42, n_estimators=20)
lgbm_clf = lightgbm.LGBMClassifier(random_state=42, n_estimators=20)
catboost_clf = catboost.CatBoostClassifier(random_state=42, verbose=0, n_estimators=20)
hist_gb_clf = HistGradientBoostingClassifier(random_state=42)
```

W tym przykładzie postanowiłem zautomatyzować uczenie i testowanie. Cały ten proces został umieszczony w funkcji evaluate_model

```
# Zdefiniuj funkcję do oceny modelu i zwrócenia wyników
def evaluate_model(clf, X_train, y_train, X_test, y_test):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred, )
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')
    auc = roc_auc_score(y_test, clf.predict_proba(X_test), average='weighted', multi_class='ovr')
    return accuracy, recall, f1, auc
```

Następnie definiuję tablicę z kalsyfikatorami, w pentli for poruszam się po każdym elemencie tablicy - każdym klasyfikatorze i wywołuję funkcję evaluate_model z odpowiednimi parametrami. Zwrócone dane zostają zapisane do obiektu results.

```
for model in models:
    accuracy, recall, f1, auc = evaluate_model(model, X_train, y_train, X_test, y_test)
    results['Method'].append(model.__class__.__name__)
    results['ACC'].append(accuracy)
    results['Recall'].append(recall)
    results['F1'].append(f1)
    results['AUC'].append(auc)
```

Na podstawie zgromadzonych danych zostaje utworzony DataFrame i zapisany do pliku xlsx.

```
# Tworzenie ramki danych z wynikami
results_df = pd.DataFrame(results)

# Zapisanie wyników do pliku Excel (xlsx)
results_df.to_excel('comparison.xlsx', index=False)
```

Widok zapisanego pliku comparison.xlsx.

	A	B	C	D	E
1	Method	ACC	Recall	F1	AUC
2	RandomForest	0.91	0.91	0.91	0.99
3	AdaBoostClass	0.52	0.52	0.39	0.87
4	GradientBoost	0.86	0.86	0.86	0.96
5	BaggingClassif	0.91	0.91	0.91	1.00
6	ExtraTreesClas	0.91	0.91	0.91	0.99
7	LGBMClassifie	0.90	0.90	0.90	0.99
8	CatBoostClass	0.85	0.85	0.85	0.98
9	HistGradientB	0.37	0.37	0.34	0.63

3. Wnioski

Uczenie zespołowe to technika w dziedzinie uczenia maszynowego, która polega na łączeniu wyników wielu modeli, aby uzyskać lepsze wyniki predykcyjne niż pojedynczy model. Jest to jak "gromadzenie mądrości grupy" - poprzez konsultację wielu ekspertów, można dokładniej przewidzieć wyniki.
Głównym celem uczenia zespołowego jest poprawienie jakości predykcji modelu poprzez łączenie wyników wielu modeli bazowych
Poprzez łączenie wyników wielu modeli, uczenie zespołowe może pomóc w zwiększeniu dokładności predykcji. Wielu "ekspertów" pracujących razem może osiągnąć lepsze wyniki niż pojedynczy ekspert.
Redukcja wariancji: Modele bazowe mogą mieć różne błędy i odchylenia od poprawnej prognozy. Poprzez łączenie wyników, model jest bardziej stabilny i mniej podatny na overfitting.
Uczenie zespołowe może poprawić zdolność modelu do generalizacji na nowe dane, co jest kluczowe w zastosowaniach praktycznych

Podczas laboratorium używaliśmy 4 miar oceny klasyfikatora:

Dokładność mierzy ogólną zdolność klasyfikatora do poprawnego przewidywania wszystkich przypadków.

$$accuracy = \frac{liczba\ poprawnie\ sklasyfikowanych\ przypadków}{całkowita\ liczba\ przypadków}$$

Precyzja mierzy zdolność klasyfikatora do poprawnego wykrywania pozytywnych przypadków.

$$precision = \frac{Liczba\ prawdziwie\ pozytywnych\ przypadków}{Liczba\ prawdziwie\ pozytywnych\ przypadków + Liczba\ fałszywie\ pozytywnych\ przypadków}$$

Recall mierzy zdolność klasyfikatora do wykrywania wszystkich rzeczywistych pozytywnych przypadków.

$$recall = \frac{Liczba\ prawdziwie\ pozytywnych\ przypadków}{Liczba\ prawdziwie\ pozytywnych\ przypadków + Liczba\ fałszywie\ negatywnych\ przypadków}$$

F1-score jest miarą, która łączy zarówno precyzję, jak i recall w jedną miarę. Jest szczególnie użyteczna w przypadku problemów z niezrównoważonymi danymi.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

AUC - (Area Under the Receiver Operating Characteristic Curve) to metryka oceny jakości klasyfikatorów binarnych i wieloklasowych. Pomaga ocenić zdolność klasyfikatora do rozróżniania między klasami poprzez analizę krzywej ROC (Receiver Operating

Characteristic). Wyższy wynik ROC AUC oznacza lepszą jakość klasyfikatora, jeśli wynik jest powyżej 0,5, to klasyfikator jest lepszy niż losowy.

VotingClassifier umożliwia łączenie wyników wielu różnych klasyfikatorów w jednym modelu. Szczególnie stosowany w technikach uczenia zespołowego, w którym to wynik jest określany na podstawie większościowego głosowania lub ważonego głosowania.
Hard Voting (Głosowanie większościowe): uzyskuje największą liczbę głosów (czyli większość) jest wybierany jako ostateczna decyzja modelu zespołowego.
Soft Voting (Głosowanie ważone): Wszystkie modele bazowe przewidują prawdopodobieństwa przynależności do klas, a VotingClassifier sumuje te prawdopodobieństwa ważone wagami i wybiera klasę z najwyższym ważonym prawdopodobieństwem.

RandomForest tworzy wiele drzew decyzyjnych, a następnie łączy wyniki ich głosowania, co prowadzi do modelu o wysokiej odporności na overfitting i zdolności do obsługi dużej ilości danych.
AdaBoost tworzy sekwencję słabych klasyfikatorów i nadaje wagę błędnie sklasyfikowanym przykładom. Każdy nowy klasyfikator jest dostosowywany do błędów poprzednich, co prowadzi do modelu o dużej zdolności do poprawnego klasyfikowania trudnych przypadków.
Gradient Boosting metoda w której model jest tworzony sekwencyjnie poprzez minimalizowanie gradientu funkcji straty. Pozwala to na budowę modelu o dużej zdolności do generalizacji i dobrze radzącego sobie z problemami klasyfikacji i regresji.
Bagging (Bootstrap Aggregating) to technika, w której wiele niezależnych klasyfikatorów jest trenowanych na losowo wygenerowanych podzbiorach danych treningowych.
Extra Trees to wariant algorytmu Random Forest, w którym drzewa decyzyjne są tworzone na podstawie losowych podzbiorów cech i próbek. Jest bardziej losowy niż Random Forest, co pomaga w zmniejszeniu wariancji modelu.
LightGBM wydajny algorytm uczenia zespołowego oparty na drzewach decyzyjnych. Wykorzystuje techniki przetwarzania gradientowego i histogramowego, co pozwala na szybsze trenowanie i uzyskanie modeli o wysokiej dokładności.
CatBoost został zaprojektowany specjalnie do obsługi danych kategorycznych.
HistGradientBoosting opiera się na drzewach decyzyjnych i wykorzystuje histogramy do przyspieszenia trenowania.