Recenzja filmów C.D.

Do wykonania zadania należy wykorzystać kod i dane z poprzedniego ćwiczenia. Poprawnie działający kod powinien wyglądać następująco:

```
import pandas as pd
from sklearn.utils import shuffle


# Wczytaj dane
with open('positive.txt', 'r', encoding='utf-8') as file:
    positive_data = file.readlines()

with open('negative.txt', 'r', encoding='utf-8') as file:
    negative_data = file.readlines()

# Przydziel klasy (0 - positive, 1 - negative)
positive_df = pd.DataFrame({'text': positive_data, 'class': 0})
negative_df = pd.DataFrame({'text': negative_data, 'class': 1})

# Połącz ramki danych
df = pd.concat([positive_df, negative_df], ignore_index=True)

# Przemieszaj zbiór danych
df = shuffle(df)

print(df.head())
```

Cel ćwiczenia:

Opracować model klasyfikacji recenzji filmu.

Aby osiągnąć cel, należy przeprowadzić testy kilku klasyfikatorów (np. Drzewa decyzyjne, Random Forest, SVM). Do ich przeprowadzenia użyjemy funkcji narzędzia *pipeline*

1. Zaimportuj wszystkie potrzebne biblioteki

```
import pandas as pd
from sklearn.utils import shuffle
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
```

2. Podziel dane na zbór uczący i testowy (Ważne!!! Zbioru testowego nie można zmieniać w trakcie badań). Zbiór testowy będzie złożony z 20% całości danych (test_size)

```
# Podziel dane na zbiór treningowy i testowy
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['class'],
test_size=0.2, random_state=42)
```

3. Utwórz pipeline klasyfikatorów

```
classifiers = [
    ('Decision Tree', DecisionTreeClassifier()),
    ('Random Forest', RandomForestClassifier(n_estimators=100,
random_state=42)),
    ('SVM', SVC())
]
```

4. Utwórz pętlę która przetestuje wszystkie klasyfikatory

```
results = []

    for classifier_name, classifier in classifiers:
        # Utwórz pipeline z CountVectorizer i klasyfikatorem
        pipeline = Pipeline([
            ('vectorizer', CountVectorizer()),
            ('classifier', classifier)
        ])

        cv_scores = cross_val_score(pipeline, X_train, y_train, cv=5,
scoring='accuracy')

        # Trenuj model
        pipeline.fit(X_train, y_train)

        # Przewiduj na danych testowych
        y_pred = pipeline.predict(X_test)

        # Oceniaj wyniki
        accuracy = accuracy_score(y_test, y_pred)
        report = classification_report(y_test, y_pred)

        # Dodaj wyniki do listy
        results.append({
            'Classifier': classifier_name,
            'Mean Accuracy': cv_scores.mean(),
            'Cross-Validation Scores': cv_scores,
            'Classification Report': report
        })
```

5. Wyświetl wyniki

```
for result in results:
    print(f"Classifier: {result['Classifier']}")
    print(f"Cross-Validation Scores: {result['Cross-Validation Scores']}")
    print(f"Mean CV Accuracy: {result['Mean Accuracy']:.4f}")
    print("Classification Report:")
    print(result['Classification Report'])
    print("=" * 50)
```

6. Zmodyfikuj kod w taki sposób, aby przetestować usuwanie krótkich wyrazów ze słownika BoW Utwórz listę vectorisers (na wzór *classifiers* patrz pkt 3) w której umieścisz różne kombinacje tworzenia słownika np.

```
CountVectorizer(token_pattern=r'\b\w{3,}\b')
```

Następnie zmodyfikuj pętlę z punktu 4 oraz wyświetlanie wyników, tak aby otrzymać raporty z każdej kombinacji

Zastanów się który algorytm działa najlepiej!