Raport - Naive Bayes

Ogólny opis

Program ten jest implementacją klasyfikatora Bayesa, jest on napisany w języku programowania Python. Wykorzystując plik z danymi klasyfikuje dowolne przypadki rekordów z danych wykorzystując właśnie klasyfikator Bayesowski. Przechodzi przez 10 takich cykli na bieżąco wyliczając dokładność algorytmu, na koniec wylicza z tego średnią oraz odchylenie standardowe.

Pliki

Folder *files* służy do przechowywania wszystkich niezbędnych danych. Aby program zadziałał, należy umieścić plik z danymi w folderze *files.* Na skutek działania programu folder ten wypełnia dodatkowo 5 plików. Kolejno:

- **after.csv** - Plik ten został specjalnie sformatowany do .csv, aby umożliwić wizualizację danych za pomocą pandas oraz matplotlib. Znajdują się tam

dane **po** klasyfikacji, z danymi potencjalnie niepoprawnymi. Tworzony pod warunkiem użycia funkcji show_graph().

- **before.csv** - Plik ten został specjalnie sformatowany do .csv, aby umożliwić wizualizację danych za pomocą pandas oraz matplotlib. Znajdują się tam dane **przed** klasyfikacji, z danymi potencjalnie niepoprawnymi. Tworzony pod warunkiem użycia funkcji show_graph().

- **cars_evaluation.trn** - Plik ten jest częścią oryginalnych danych. Znajduje się w nim 70% danych i są one przeznaczone do treningu.

- **cars_evaluation.tst** - Plik ten jest częścią oryginalnych danych. Znajduje się w nim 30% danych i są one przeznaczone do testów.

- **cars_evaluation_format.tst** - Plik ten jest częścią oryginalnych danych. Znajduje się w nim 30% danych i są one przeznaczone do klasyfikacji. W praktyce oznacza to, że ich klasyfikacja została wycięta.

Najważniejsze funkcje

def show_graph():

Funkcja ta została zaimplementowana, ponieważ w dobry sposób wizualizuje dane pierwotne, oraz świeżo zaklasyfikowane. Przedstawia je w formie dwóch grafów.

def handle_files():

Funkcja ta obsługuje pierwotny plik danych, rozdzielając go na dwa osobne, tzn. testowy oraz treningowy w proporcjach 70/30. Przy tym tworzy dodatkowy plik który zostanie poddany klasyfikacji. Jest on potrzebny, aby porównać z sklasyfikowanymi już wcześniej danymi.

def calculate_probability(car, keyword):

- car - pojedynczy rekord danych
- keyword - klasyfikator (unacc, acc, good, vgood)

Funkcja ta na podstawie podanego pojedynczego rekordu danych oraz klasyfikatora oblicza prawdopodobieństwo przynależności do danej klasy. Dodatkowo zostało zaimplementowane przekształcenie Laplace'a, aby wykluczyć przypadki w których licznik/mianownik jest równy 0. Do tego wykorzystywane są poniższe trzy funkcje. Różni je jedynie sytuacyjność.

Służą one zliczaniu pojedynczych anotacji. Jedyną różnicą między nimi jest poniższy if:

def check_both(w1, w2, i):

```
if w1 in c.split(',')[i] and w2 in
c.split(',')[6]:
```

def count_specific_word(w):

```
if w in c.split(',')[i]:
```

def count_word(w):

```
if w in c:
```

Wyniki

Wyniki po 10 cyklach losowania danych oraz ich klasyfikowania wyglądają następująco:

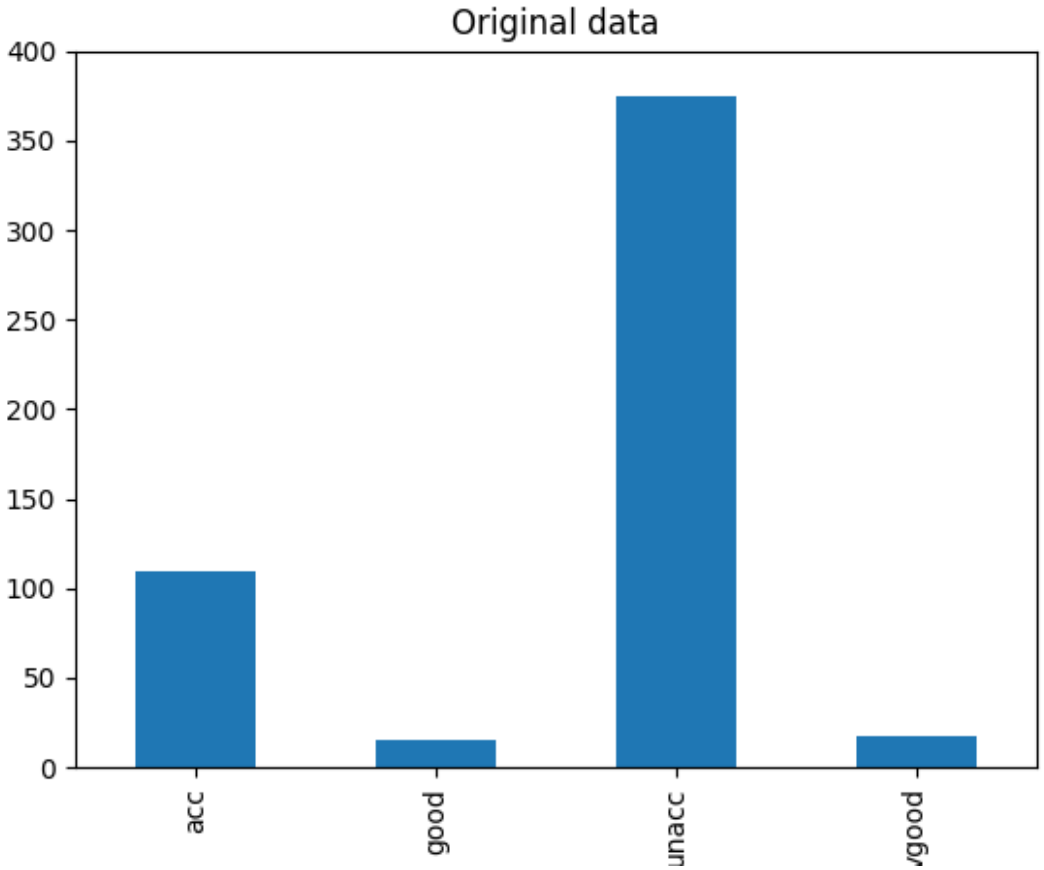
- Cykl 0 - Dokładność 91%
- Cykl 1 - Dokładność 91%
- Cykl 2 - Dokładność 93%
- Cykl 3 - Dokładność 92%
- Cykl 4 - Dokładność 92%
- Cykl 5 - Dokładność 95%

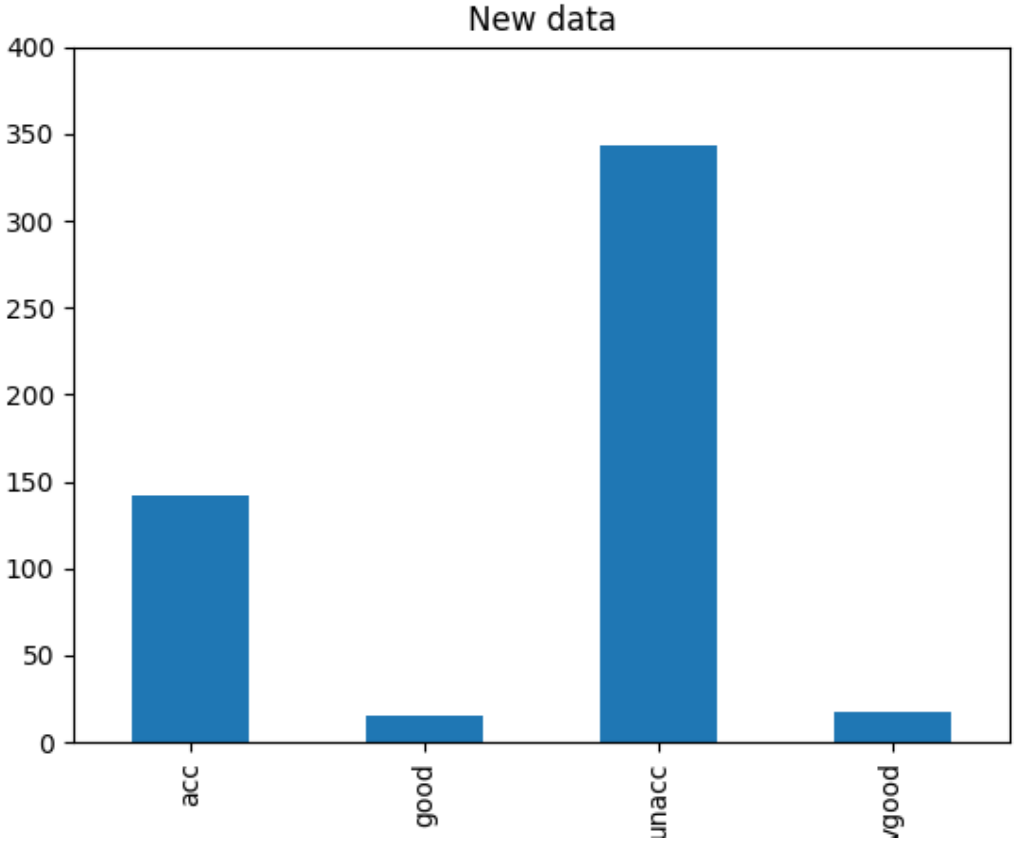
- Cykl 6 - Dokładność 93%
- Cykl 7 - Dokładność 92%
- Cykl 8 - Dokładność 92%
- Cykl 9 - Dokładność 91%

Średnia przez 10 cykli: 92%
Odchylenie standardowe: 1,18321

Wnioskując, wyniki otrzymują się w granicach 91-95%, stąd odchylenie standardowe jest niskie, co oznacza, że dane są skupione wokół średniej.

Poniżej przedstawiam dwa grafy gdzie dokładność klasyfikacji wyniosła 93%.


Original data
400
350
300
250
200
150
100
50
0
acc
good
unacc
vgood


New data
400
350
300
250
200
150
100
50
0
acc
good
unacc
vgood