**Въведение в Машинното обучение**

**Лабораторно упражнение №7**

**Класификация на текст**

**Алгоритми за класификация в машинното обучение.**

1. **Цел на упражнението.**

Целта на лабораторното упражнение е студентите да усвоят темата класификация на текст, като я приложат при решаване на класификационни задачи.

1. **Алгоритми за класификация.**

**Постановка на задачата:**

Нека:

* Х∈Rn – множество на обектите (входни данни)
* У∈R - множество на резултатите (изходни данни)

Ще разгледаме двойката (*х, у*), като реализация на (n+1) – мерни случайни величини (Х,У), зададени във вероятностното пространство.

Законът на разпределение РХУ(*х, у*) не е известен, известна е само обучаваща извадка:

{*х*(1)*, у*(1*), х*(2)*, у*(2)*, ……, х*(*N*)*, у*(*N*)} (1)

където се явяват независими реализации на случайна величина .

Целта е да се намери функция *,* която изхождайки от стойностите на *х*, да предскаже *у*. Функцията *f* се нарича решаваща функция или класификатор. Построяването на *f* наричаме обучение, настройки на модели и т.н.

* 1. **Наивен Бейсов класификатор**

Методът на Бейс, използван в машинното обучение се основава на известната формула на Бейс:

,  (2)

за определяне на постериорната вероятност за настъпването на събитието А съвместно с хипотезата *Hk*.

Във формулата на Бейс:

* *А* е случайно събитие, настъпването на което винаги се съпътства с едно от събитията , наричани хипотези и образуващи пълна група от несъвместими събития;
*   е априорната вероятност на хипотезата  ;
*  е условната вероятност за настъпване на събитието *А* при условие, че е настъпила хипотезата ;
*  е вероятността за настъпване на събитието *А*, която се изразява с формулата за пълната вероятност:



* + 1. **Мултиноминален Бейсов класификатор**

Мултиноминалният Наивен Бейсов класификатор, прави предположението, че признаците са разпределени мултиноминално.

Нека xi∈{1,…K}, имат вероятности за поява θ1, …θК

Тогава, вероятността за поява на събитието х при даден признак θ е:

където:

Многономиналният Наивен Бейсов класификатор изчислява честотата на поява на всяка дума в документите. Отново се прави наивното предположение, че вероятността дадена дума да се среща в текста е независима от контекста и от позицията на думата в документа.

**Задача:** Да се предскаже изхода от играта на отбор при дадени данни от предходни мачове (Таблица I).

Таблица I

Данни от предходни мачове

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Време** | **Терен** | **Влажност** | **Ветровито** | **Ключови играчи** | **Изход от двубой при мач** |
|  | **Х1** | **Х2** | **Х3** | **Х4** | **Х5** | **У** |
|  | Слънчево | Свой | Висока | НЕ | Не присъстват | Победа |
|  | Слънчево | Свой | Висока | ДА | Присъстват | Победа |
|  | Облачно | Чужд | Ниска | НЕ | Присъстват | Победа |
|  | Дъждовно | Чужд | Висока | НЕ | Присъстват | Равенство |
|  | Слънчево | Свой | Ниска | НЕ | Не присъстват | Победа |
|  | Дъждовно | Чужд | Ниска | ДА | Не присъстват | Загуба |
|  | Облачно | Свой | Ниска | ДА | Присъстват | Победа |
|  | Слънчево | Чужд | Висока | НЕ | Не присъстват | Загуба |
|  | Слънчево | Чужд | Ниска | НЕ | Присъстват | Равенство |
|  | Облачно | Свой | Ниска | НЕ | Присъстват | Победа |

Априорна вероятност на У се изчислява като отношението на брой благоприятни изходи към общия брой. В примера общия брой записи е 10, победите са 6, загубите са 2, както и равенството от играта (ТаблицаII).

Таблица II

Априорна вероятност

|  |  |  |  |
| --- | --- | --- | --- |
|  | Победа | Загуба | Равенство |
| Априорна вероятност на У | 0.6 |  |  |

Следва преброяване на победите, загубите и равенството при слънчево, облачно и дъждовно време (Таблица III).

Таблица III

Честотна таблица

|  |  |  |  |
| --- | --- | --- | --- |
| **Х1** | **У** | | |
| Време | Победа | Загуба | Равенство |
| Слънчево | 3 |  |  |
| Облачно | 3 |  |  |
| Дъждовно | 0 |  |  |
| **Общо** | **6** |  |  |

Въз основа на Таблица III се изчислява условната вероятност за победи, загубите и равенството при слънчево, облачно и дъждовно време като отношението на брой благоприятни изходи към общия брой (Таблица IV).

Таблица IV

Условна вероятност

|  |  |  |  |
| --- | --- | --- | --- |
| **Условна вероятност** | | | |
| **Х1** | **У** | | |
| Време | Победа | Загуба | Равенство |
| Слънчево | 3/6=1/2 |  |  |
| Облачно | 3/6=1/2 |  |  |
| Дъждовно | 0 |  |  |

Следва преброяване на победите, загубите и равенството при различен терен, който може да бъде чужд или свой (Таблица V).

Таблица V

Честотна таблица

|  |  |  |  |
| --- | --- | --- | --- |
| **Х2** | **У** | | |
| **Терен** | Победа | Загуба | Равенство |
| Свой | 5 |  |  |
| Чужд | 1 |  |  |
| **Общо** | **6** |  |  |

Въз основа на Таблица V се изчислява условната вероятност за победи, загубите и равенството при различен терен, който може да бъде чужд или свой като отношението на брой благоприятни изходи към общия брой (Таблица VI).

Таблица VI

Условна вероятност

|  |  |  |  |
| --- | --- | --- | --- |
| **Условна вероятност** | | | |
| **Х2** | **У** | | |
| **Терен** | Победа | Загуба | Равенство |
| Свой | 5/6 |  |  |
| Чужд | 1/6 |  |  |

Следва преброяване на победите, загубите и равенството при различна влажност (Таблица VII).

Таблица VII

Честотна таблица

|  |  |  |  |
| --- | --- | --- | --- |
| **Х3** | **У** | | |
| **Влажност** | Победа | Загуба | Равенство |
| Висока | 2 |  |  |
| Ниска | 4 |  |  |
| **Общо** | **6** |  |  |

Въз основа на Таблица VII се изчислява условната вероятност за победи, загубите и равенството при различна влажност като отношението на брой благоприятни изходи към общия брой (Таблица VIII).

Таблица VIII

Условна вероятност

|  |  |  |  |
| --- | --- | --- | --- |
| **Условна вероятност** | | | |
| **Х3** | **У** | | |
| **Влажност** | Победа | Загуба | Равенство |
| Висока | 2/6=1/3 |  |  |
| Ниска | 4/6=2/3 |  |  |

Следва преброяване на победите, загубите и равенството при наличие или не наличие на вятър (Таблица IX).

Таблица IХ

Честотна таблица

|  |  |  |  |
| --- | --- | --- | --- |
| **Х4** | **У** | | |
| **Ветровито** | Победа | Загуба | Равенство |
| ДА | 2 |  |  |
| НЕ | 4 |  |  |
| **Общо** | **6** |  |  |

Въз основа на Таблица IX се изчислява условната вероятност за победи, загубите и равенството при при наличие или не наличие на вятър като отношението на брой благоприятни изходи към общия брой (Таблица X).

Таблица Х

Условна вероятност

|  |  |  |  |
| --- | --- | --- | --- |
| **Условна вероятност** | | | |
| **Х4** | **У** | | |
| **Ветровито** | Победа | Загуба | Равенство |
| ДА | 2/6=1/3 |  |  |
| НЕ | 4/6=2/3 |  |  |

Следва преброяване на победите, загубите и равенството при наличие или не наличие на ключови играчи за отбора (Таблица XI).

Таблица ХI

Честотна таблица

|  |  |  |  |
| --- | --- | --- | --- |
| **Х5** | **У** | | |
| **Ключови играчи** | Победа | Загуба | Равенство |
| Присъстват | 4 |  |  |
| Не присъстват | 2 |  |  |
| **Общо** | **6** |  |  |

Въз основа на Таблица XI се изчислява условната вероятност за победи, загубите и равенството при при наличие или не наличие на вятър като отношението на брой благоприятни изходи към общия брой (Таблица XII).

Таблица ХII

Условна вероятност

|  |  |  |  |
| --- | --- | --- | --- |
| **Условна вероятност** | | | |
| **Х5** | **У** | | |
| **Ключови играчи** | Победа | Загуба | Равенство |
| Присъстват | 4/6=2/3 |  |  |
| Не присъстват | 2/6=1/3 |  |  |

Следва приложение на формулата на Бейс.

**Задача**: Да се предскаже изхода от играта на отбор при дадени данни от предходни мачове (Таблица I).

Във формула (6) думата равен. е съкращение на равенство.

В последните три формули знаменателите са равни, затова може да се пропуснат при изчисленията.

Ако

*Р1=* (7)

*Р2=* (8)

*Р*3= (9)

*Р1=*(10)

*Р2*  (11)

*Р*3 (12)

Резултатите в последните две формули са нули, затова се прилага Лапласово изглаждане, при което се добавя 1 в числителя, а в знаменателя сумата от броя на значенията на отделните характеристики.

Таблица ХIII

Условна вероятност

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Условна вероятност** | | | | | |
| **Х1** | **У** | | | | |
| Време | Победа | | | Загуба | Равенство |
| Слънчево | (3+1)/(6+3)=4/9 | | |  |  |
| Облачно | (3+1)/(6+3)=4/9 | | |  |  |
| Дъждовно | (0+1)/(6+3)=1/9 | | |  |  |
| **Условна вероятност** | | | | | |
| **Х2** | **У** | | | | |
| **Терен** | Победа | | | Загуба | Равенство |
| Свой | (5+1)/(6+2)=6/8=3/4 | | |  |  |
| Чужд | (1+1)/(6+2)=2/8=1/4 | | |  |  |
| **Условна вероятност** | | | | | |
| **Х3** | **У** | | | | |
| **Влажност** | Победа | | | Загуба | Равенство |
| Висока | (2+1)/(6+2)=3/8 | | |  |  |
| Ниска | (4+1)/(6+2)=5/8 | | |  |  |
| **Условна вероятност** | | | | | |
| **Х4** | | **У** | | | |
| **Ветровито** | | Победа | | Загуба | Равенство |
| ДА | | (2+1)/(6+2)=3/8 | |  |  |
| НЕ | | (4+1)/(6+2)=5/8 | |  |  |
| **Условна вероятност** | | | | | |
| **Х5** | | | **У** | | |
| **Ключови играчи** | | | Победа | Загуба | Равенство |
| Присъстват | | | (4+1)/(6+2)=5/8 |  |  |
| Не присъстват | | | (2+1)/(6+2)=3/8 |  |  |

При предсказване на У (изход от двубой при мач), ако

Х1 (времето) = Слънчево,

Х2 (терен) = Свой,

Х3 (влажност) = Ниска,

Х4 (ветровито) = НЕ,

Х5 (ключови играчи) = Присъстват

### Заместват се стойностите от Таблица ХIII във формули (10), (11) и (12)

*Р1=*  (13)

*Р2*  (14)

*Р3* (15)

**Задача 1:** Изчислете самостоятелно формулите, като използвате предоставените данни. Отговорете на въпроса, коя е най-голямата вероятност: отборът да спечели, да загуби или да завърши с равенство (3 точки).

От формули (13), (14) и (15) следва, че Р…..>P……>P……, т.е. най-голяма е вероятността отборът …………………………………………...

* + 1. **Бернулиев Бейсов класификатор**

Бернулиевият Бейсов класификатор представя документа, като бинарен вектор на пространството. Стойност единица, показва дали думата се появява поне веднъж в документа. С такова представяне, правим наивното предположение на Бейс, че вероятността всяка дума да се среща в текста е независима от появата на други думи.

Нека x∈[0,1] е бинарна променлива, която има вероятност за поява θ.

Тогава:

* 1. **Метод на опорните вектори - SVM (Support Vector Mashine)**

Методът на опорните вектори (SVM, Support Vector Machines) представя обучаващите примери като точки в n-мерно пространство. Примерите са проектират в пространството по такъв начин, че да бъдат линейно разделими. При работа с два класа се търси начин да се начертае линия, която да разделя данните от двата класа. Линията, която разделя данните, се нарича разделителна хипер равнина. Тази хипер равнина трябва да се избере по такъв начин, че да се намира възможно най-далеч от примерите и на двата класа.

Функцията f(x) на линейната класификация е във вида:

*f(x) = wTx + b*

където: *wТ е* тегловен вектор, а *b* е отклонението

След като се определи, къде да бъде поставена разделителната линия, се калкулира допустимата границата:

*label* \* (*wTx + b*

Целта е да се намерят стойностите на *wT* и *b*, които ще определят класификатора. За да се направи това, е необходимо да се намерят точките с най-малко отклонение, който трябва да се максимизира. Това може да се запише по този начин:

* 1. **Метод на К най-близкия съсед (КNN - k nearest neighbours)**

Методът на К най-близкия съсед е метрически алгоритъм за класификация на обекти, основан на оценка на сходство на обекти. Класифицираният обект се отнася към този клас, към който принадлежат най-близките до него обекти на обучаващата извадка (КNN - k nearest neighbours).

Алгоритъм

* Определяне на К броя обекти, които ще участват в класификацията
* Изчисляване на дистанцията между всеки два обекта от обучаващата извадка, чрез използване на подходяща функция за измерване на разстояние между две точки
* Избор на К обекти от обучаващата извадка, разстоянието до което е минимално
* Класа на класифицирания обект – това е класа на новия обект, намерен въз основа на заключенията, направени по отношение на К най-близки съседи

Видове функции, изчисляващи разстояние:

* Евклидово
* Lp – метрика
* L∞ – метрика
* L1 – метрика
* Ланс – Уилямс
* Косинусова мяра

– вектор от m-признаци на i- я обект

– вектор от m-признаци на j- я обект

Важен въпрос при работа с метода на К най-близкия съсед е изборът на числото К – това е броя на най-близките съседи. Евристични техники, като кръстосано валидиране могат да помогнат за получаването на подходящи стойности на K.

* 1. **Метод Дърво на решенията**

Дърво на решенията (класификационни дървета) (Decision trees) е метод за представяне и решаване на класификационни задачи и се отнася към логическите методи за класификация. Строят се за изграждане на модел въз основа на статистически оценки и поредици от проверки на логически условия.

За всеки възел на дървото се избира оптимално правило за разделяне, включващо избор на атрибут и стойност на този атрибут, по които да стане разделянето. Изборът на атрибут се основава на информационната печалба при разделяне на обектите по съответния атрибут.

Информационната печалба на атрибута А, по отношение на набор от примери S, се изчислява като:

където Sv е множество от примери, които имат стойност v за атрибута А, Е(S) - ентропия на S множество от примери.

Ентропията се представя в следния вид:

където

n – брой примери

рi – вероятността да се избере i клас.

* 1. **Random Forest**

Случайните гори (Random Forest) се основават на ансамблов метод за самообучение (ensemble method). Random Forest класификаторът създава набор от дървета за вземане на решения от произволно избран набор от тренировъчни комплекти. Алгоритъмът използва усредняване с цел подобряване точността на прогнозирането. Всяко отделно дърво представлява случайна извадка от обучаващото множество, като изборът е независим и използва еднакво разпределение на всички дървета. Всяко дърво осъществява класификация, след което статистическа мода на всички отделни класификации се взема като крайна класификация на гората. Обучаването става с едни и същи алгоритми, с един и същи обучаващ набор, но се прилага на различни случайни подмножества.

* 1. **AdaBoost**

Алгоритъмът AdaBoost (Adaptive Boosting) е представител на ансамбловия алгоритъм boosting. Това е итерационен алгоритъм, при който „силния“ класификатор допуска малки грешки при обучението на основата на „слабия“ класификатор, който правилно класифицира повече от 50%. В ансамбловия алгоритъм Adaboost различните класификатори се обучават последователно. Всеки нов класификатор е обучен въз основа на ефективността на вече съществуващи обучени. Усилването се състои в съсредоточаване на новите класификатори върху данни, които са били погрешно класифицирани от предишни класификатори.

1. **Библиотека Natural Language Toolkit**

Практическата работа при обработка на естествен език, обикновено използва големи обеми от лингвистични данни или корпуси. **NLTK (Natural Language Toolkit)** e open source библиотека на Python, предоставящ безплатно онлайн книги, статии и данни. [[1]](#footnote-1)

* 1. **Текстови корпуси на NLTK**
* **Корпус Гутенберг**

NLTK включва малка колекция от текстове от електронния текстов архив на Project Gutenberg, който съдържа около 25 000 безплатни електронни книги, които се намират на <http://www.gutenberg.org/>

**>>> import** **nltk**

**>>>** **from nltk.corpus import gutenberg**

* **Уеб и чат текстове**

Малка колекция от уеб текстове на NLTK включва съдържание от дискусионен форум на Firefox, разговори, които са чути в Ню Йорк, филмовия скрипт на Carribean Pirates, лични реклами и рецензии за вино.

**>>>** **from nltk.corpus import webtext**

Съществува и корпус от чат-сесии от незабавни съобщения, първоначално събрани от Военноморското училище за следдипломна квалификация за изследване на автоматичното откриване на интернет хищници. Корпусът съдържа над 10 000 публикации, като са заменени потребителските имена с общи имена "UserNNN" и ръчно редактирани, за да премахната идентифицираща информация. Корпусът е организиран в 15 файла, където всеки файл съдържа няколкостотин публикации, събрани за дадена дата, за възрастова група (тийнейджъри, 20, 30 и 40, както и общ чат за възрастни). Името на файла съдържа датата, чата и броя публикации; напр. 10-19-20s\_706posts.xml съдържа 706 публикации, събрани от 20-те години на чат на 10/19/2006.

**>>> from nltk.corpus import nps\_chat**

* **Корпус Браун**

Браун Corpus е първият милион електронен корпус на английски, създаден през 1961 г. в университета Браун. Този корпус съдържа текст от 500 източника и източниците са категоризирани по жанр, като: *новини*, *мнения, хобита, мистерии, научна фантастика, хумор и други*. Пълен списък на всички раздели, може да видите на следния адрес:<http://icame.uib.no/brown/bcm-los.html>

**>>>** **from nltk.corpus import brown**

* **Корпус Ройтерс**

Reuters Corpus съдържа 10,788 новинарски документа, възлизащи на 1,3 милиона думи. Документите са класифицирани в 90 теми и групирани в два набора, наречени "обучение" и "тест".

**>>> from nltk.corpus import reuters**

* **Корпус: Inaugural Address Corpus**

Колекция от 55 текста, известни като президентските уводни речи на САЩ от 1789 година до 2009 година, събрани и предоставени от Kathleen Ahrens.

**>>>** **from nltk.corpus import inaugural**

* **Корпус: mouvie\_review**

Колекция от 1000 положителни и 1000 отрицателни отзиви на филми, използвани за анализ на полярността на настроенията, предоставени през 2004 година от Bo Pang и Lillian Lee**.**

**>>>** **from nltk.corpus import movie\_reviews**

* **Корпус: names**

Колекция от 5001 английски женски имена и 2943 мъжки имена, сортирани по азбучен ред, предоставени през 1991 година от Mark Kantrowitz.

* **Зареждане на собствен корпус**

Ако имате своя собствена колекция на текстови файлове, можете да ги заредите с помощта на NLTK на PlaintextCorpusReader.

**>>> from nltk.corpus import PlaintextCorpusReader**

* 1. **Структура на текстовия корпус**

Най-простият вид корпус е колекция от текстове без конкретна организация. Често текстовете се групират в категории, които биха могли да съответстват на жанра, източника, автора, езика и т.н. Понякога тези категории се припокриват, особено в случаите на актуални категории, тъй като текстът може да е релевантен за повече от една тема. Повече документация може да се намери с помощта на nltk.corpus.reader или <http://nltk.org/howto>.

1. **Изграждане на класификационни модели с Pyhon**

За да се изгради модел на класификатор в Python, може да се използва библиотеката scikit-learn (съкращение за "Scientific Kit for Learning"), която предлага различни модели за машинно обучение.

Класът SklearnClassifier от библиотеката scikit-learn в Python е предназначен да бъде използван за прилагане на различни класификационни модели в средата на NLTK (Natural Language Toolkit). Като основен метод, той използва метода fit() на обектите от класовете scikit-learn за обучение на моделите.

SklearnClassifier е наследник на базовия клас ClassifierI в NLTK и за да се използва, трябва да се импортира от модула nltk.classify.scikitlearn. След това може да се създаде обект от класа SklearnClassifier и да се подаде някой от класификационните модели на scikit-learn за обучение и предсказване на данните.

* 1. **Методи на класа SklearnClassifier от библиотеката scikit-learn на Python**

**train(self, labeled\_featuresets**): Метод, който прилага обучение на предоставените етикетирани множества от признаци. Приема списък от множества от признаци и техните етикети, където етикетът е включен към множеството от признаци като последен елемент. Методът train() използва метода fit() на обекта на класификационния модел, който е предоставен като аргумент при създаването на обекта на SklearnClassifier. След като методът train() приключи успешно, моделът ще бъде обучен и готов за предсказване на нови множества от признаци.

**classify(self, featureset):** Метод, който прилага обучения модел върху множеството от признаци и връща предсказания клас.

**accuracy(self, gold):** се използва за измерване на точността на класификационния модел, който е обучен с помощта на метода train().

Методът приема етикети на множеството от признаци, за които се знае точният клас (ground truth labels), като аргумент. Тези етикети са обикновено разделени на две групи - обучаващо множество и тестващо множество. Методът сравнява предсказаните етикети с верните етикети (ground truth labels) и измерва колко често предсказанията на модела съвпадат с верните етикети.

* 1. **Алгоритми в Python**

а) Бернулиев Бейсов класификатор

<https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html>

б) Мултиноминален Бейсов класификатор

<https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html>

в) Метод на опорните вектори с линейно ядро – (LinearSVM (Linear Support Vector Mashine))

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html>

г) Метод на К най-близкия съсед (КNN - k nearest neighbours)

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

д) Метод Дърво на решенията (Decision trees)

<https://www.nltk.org/_modules/nltk/classify/decisiontree.html>

е) Алгоритъм Случайни гори (Random Forest)

<https://scikit-learn.org/stable/modules/ensemble.html>

ж) Алгоритъм AdaBoost (Adaptive Boosting)

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>

* 1. **Допълнителни пояснения към кода на задача 2**

**а) за ред 9**

Корпусът "movie\_reviews" е част от библиотеката за обработка на естествен език NLTK (Natural Language Toolkit) за Python. Този корпус съдържа 2000 рецензии на филми, които са маркирани като положителни или отрицателни.

Рецензиите са събрани от интернет и всяка една от тях е прочетена и класифицирана от хора като положителна или отрицателна. След това, текстът на рецензиите е обработен и е премахната частта, която не е свързана с мнението на рецензента за филма.

Всяка рецензия в корпуса "movie\_reviews" е разделена в папки "pos" и "neg", в зависимост от класификацията й като положителна или отрицателна. Корпусът също така съдържа и допълнителна информация, като име на файл, рейтинг на филма и др.p

**б) за ред 13**

Този код създава списък на документи, където всеки документ представлява ревю за филм от колекцията Movie Reviews Corpus на NLTK (Natural Language Toolkit). Всеки документ е представен като списък от думи от съответния файл и съответства на определена категория (положителни или отрицателни рецензии).

По-конкретно, за всеки тип категория (положителни и отрицателни рецензии) се извличат всички файлове, които съответстват на тази категория, и се образува кортеж от списък от думите във файла и категорията на файла. В резултат, списъкът "docs" съдържа всички документи в колекцията Movie Reviews Corpus, като всеки документ е кортеж от списък от думи и категория. (Кортеж (tuple на английски език) е тип данни в програмирането, който представлява наредена редица от елементи. Кортежите са подобни на списъците, но са неизменяеми, което означава, че след като един път бъдат създадени, техните стойности не могат да бъдат променяни.)

В този код, "movie\_reviews" е обектът, който представлява Movie Reviews Corpus на NLTK. "fileid" е идентификаторът на всеки файл в колекцията, "category" е категорията на рецензията ("pos" за положителна рецензия и "neg" за отрицателна рецензия), а "docs" е списъкът, който съдържа всички документи в колекцията.

**в) от ред 27 до 32**

Този код дефинира функцията "find\_features", която приема списък от думи в документ и връща речник, който съдържа за всяка дума от множеството "word\_features" стойност True или False в зависимост от това дали думата се среща в документа или не.

След това, към всеки документ в списъка "docs" се прилага функцията "find\_features", като се създава нов кортеж, който съдържа речника с особеностите на документа и категорията му. Този процес се извършва чрез използване на генератор на списъци и резултатът е списък от кортежи, наречен "featuresets".

В резултат, "featuresets" представлява множество от кортежи, където всеки кортеж представлява един документ и съответните му особености (feature set) и категория (pos/neg). Този списък от кортежи може да се използва като данни за обучение на машинно обучение алгоритми, за да се класифицират нови документи в положителни или отрицателни рецензии.

**г) за ред 33**

В кода featuresets = [(find\_features(rev),category) for (rev,category) in docs] променливата rev се използва за представяне на текущия документ, който се обработва в момента от for цикъла.

По-точно, променливата rev получава стойността на list(movie\_reviews.words(fileid)) за всеки файл fileid в категорията category на movie\_reviews. Така за всяка категория на ревюта, програмата създава списък от всички думи в тези ревюта, като списъците за всеки ревю са запазени в кортежа docs.

В израза [(find\_features(rev),category) for (rev,category) in docs], rev и category са декомпресирани от кортежа docs посредством оператора for. Така всеки елемент rev на списъка docs е свързан със съответния елемент category от същия списък.

След като стойността на rev е получена, тя се предава на функцията find\_features(rev), за да се изготви речник от често срещани думи в този документ. Като резултат, featuresets съдържа списък от кортежи, като всеки кортеж е двойка от речника на намерените думи в дадения документ и категорията му (положително или отрицателно ревю).

**Задача 2.** **Класификация на текст**

С помощта на класификаторите: Дърво на решенията, Бернулиев Бейсов класификатор, Мултиномиален Бейсов класификатор,Метод на опорните вектори с линейно ядро, Метод на К най-близкия съсед, Ансамбловите алгоритми Random Forest и Ada Boost да се изчисли и сравни точността на разпознаване на мненията на потребителите въз основа на филмови отзиви, предоставени в корпуса: movie\_reviews. Да се визуализира точността на класификаторите с диаграма тип bar (стълбчеста). В програмата на задачата липсват части от кода, които трябва да въведете. Това са редове от 49 до 52, от 54 до 57, от 59 до 62, от 64 до 67, от 69 до 72, от 74 до 82 (2 точки).

Особености: Редовете от 38 до 42 са в коментар, защото в тази задача методът работи бавно. Махнете коментарите, когато сте готови с всички алгоритми, т.е. когато сте готови почти с цялата задача.

1. import nltk
2. from nltk.classify.scikitlearn import SklearnClassifier
3. from sklearn.svm import LinearSVC
4. from sklearn.naive\_bayes import MultinomialNB,BernoulliNB
5. from sklearn.neighbors import KNeighborsClassifier
6. from nltk.classify import DecisionTreeClassifier
7. from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier
8. import matplotlib.pyplot as plt
9. from nltk.corpus import movie\_reviews
10. import random
11. **#създаване на списък от всички документи и тяхната категория:**
12. **#neg - негативно мнение и pos - позитивно мнeние**
13. docs = [(list(movie\_reviews.words(fileid)), category)

for category in movie\_reviews.categories()

for fileid in movie\_reviews.fileids(category)]

1. **#Задаване на началната точка за генериране на псевдо-случайни числа.**
2. random.seed(43)
3. **#Разместване на случаен принцип всички документи**
4. random.shuffle(docs)
5. **#създаване на списък от всички думи и превръщането им в малки букви**
6. all\_words = []
7. for w in movie\_reviews.words():
8. all\_words.append(w.lower())
9. **#Изчисляване на честотата на срещане на всяка дума**
10. all\_words=nltk.FreqDist(all\_words)
11. **#списък с първите 3000 най-често използвани думи в корпуса**
12. word\_features = list(all\_words.keys())[:3000]
13. **#създаване на екстрактор, който проверява дали всяка от думите присъства #в даден документ**
14. def find\_features(document):
15. words = set(document)
16. features={}
17. for w in word\_features:
18. features[w] = (w in words)
19. return features
20. **featuresets = [(find\_features(rev),category) for (rev,category) in docs]**
21. **#обучаващо множество**
22. training\_set = featuresets[:1900]
23. **#тестващо множество**
24. testing\_set = featuresets[1900:]
25. **# Дърво на решенията (Decision Tree)**
26. #DT\_classifier = DecisionTreeClassifier.train(training\_set)
27. #DT\_classifier.train(training\_set)
28. #DT\_accuracy=nltk.classify.accuracy(DT\_classifier, testing\_set)\*100
29. #print("Decision Tree accuracy percent:",( DT\_accuracy))
30. **# Бернулиев Наивен Бейсов класификатор**
31. BNB\_classifier = SklearnClassifier(BernoulliNB())
32. BNB\_classifier.train(training\_set)
33. BNB\_accuracy=nltk.classify.accuracy(BNB\_classifier, testing\_set)\*100
34. print("BernoulliNB accuracy percent:",(BNB\_accuracy))
35. **#Мултиномиален Наивен Бейсов класификатор**



40. **#Метод на опорните вектори с линейно ядро**



45. **#Метод на К най-близкия к съсед**



50. **#Ансамлов алгоритъм Случайна гора (Random Forest)**



55. **#Ансамлов алгоритъм Ada Boost**



60. **# Диаграма**

**Резултат:**

Decision Tree accuracy percent: 64.0

BernoulliNB accuracy percent: 81.0

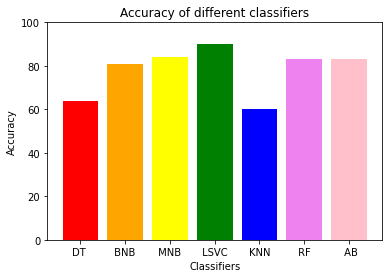
MultinomialNB accuracy percent: 84.0

LinearSVC accuracy percent: 90.0

KNN accuracy percent: 60.0

Random Forest accuracy percent: 79.0

Ada Boost accuracy percent: 83.0



1. http://www.nltk.org/book/ch02.html [↑](#footnote-ref-1)