Кафедра систем штучного інтелекту

ПОЯСНЮВАЛЬНА ЗАПИСКА

**до курсового проекту на тему:**

«Натренувати класифікатор на визначення статі за ім’ям»

Студент групи КН-309 Фарина П.Н.

**Керівник проекту** *Засоба Є.О.* (*підпис)*\_\_\_\_\_\_\_\_

**Завідувач кафедри** *Шаховська Н.Б. (підпис)*\_\_\_\_\_\_\_\_

”\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_р.

**ЗМІСТ**

Вступ………………………………………………………………………..……3

Виконання роботи……………………………………….……………….….…..5

Висновки………………………………………………………………………....8

Список літератури………………………………………………………..…...…9

Додатки………………………………………………………………………….10

Код програми…………………………………………………………………....10

**ВСТУП**

Наївний Байєсівський класифікатор один з найпростіших з алгоритмів класифікації. Проте, дуже часто він працює не гірше, а то і краще ніж більш складні алгоритми.

Розглядатися алгоритм Байєсса буде розглядатися на прикладі завдання визначення статі за іменем. Звичайно, щоб визначити стать можна створити великий список імен з мітками статі. Але цей список в будь-якому випадку буде невичерпний. Для того щоб вирішити дану проблему, можна «натренувати» модель по маркованих іменах.

**ВИКОНАННЯ РОБОТИ**

**Дослідження теми курсової роботи**.

Машинне навчання - це дослідження в галузі інформатики, штучного інтелекту та статистики. У центрі уваги машинного навчання - підготовка алгоритмів для вивчення закономірностей і прогнозування даних. Машинне навчання особливо цінно, тому що воно дозволяє використовувати комп'ютери для автоматизації процесів прийняття рішень.

Зараз існує дуже багато додатків для машинного навчання. Netflix і Amazon використовують машинне навчання для відображення нових рекомендацій. Банки використовують його для виявлення шахрайської діяльності в транзакціях з кредитними картами, а медичні компанії починають використовувати машинне навчання для моніторингу, оцінки та діагностики пацієнтів.

Байєсівський класифікатор - широкий клас алгоритмів класифікації, заснований на принципі максимуму апостеріорної ймовірності. Для класифікованого об'єкта обчислюються функції правдоподібності кожного з класів, по ним обчислюються апостеріорні ймовірності класів. Об'єкт відноситься до того класу, для якого апостеріорна ймовірність максимальна.

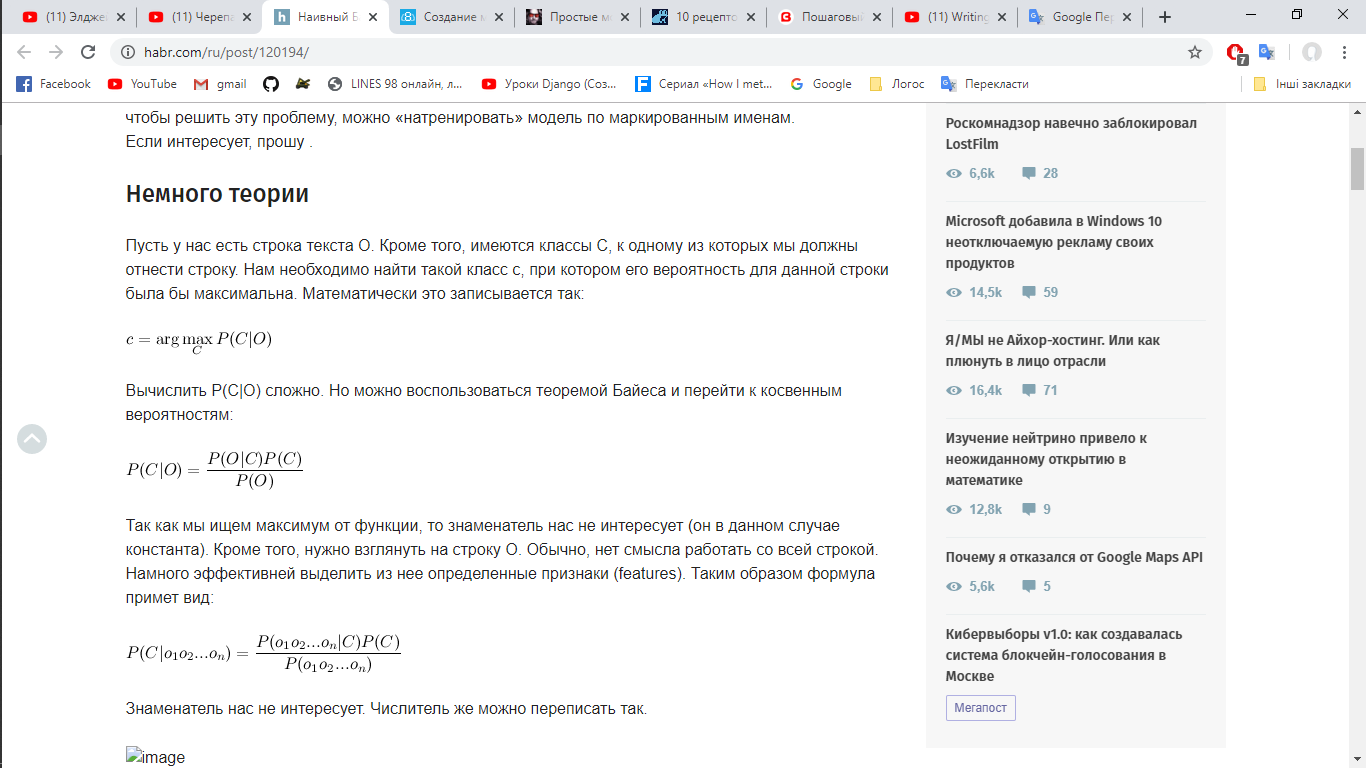
Байєсівський підхід до класифікації заснований на теоремі, яка стверджує, що якщо щільності розподілу кожного з класів відомі, то шуканий алгоритм можна виписати в явному аналітичному вигляді. Більш того, цей алгоритм оптимальний, тобто володіє мінімальною ймовірністю помилок.

На практиці щільності розподілу класів, як правило, не відомі. Їх доводиться оцінювати (відновлювати) за навчальною вибіркою. В результаті байєсівский алгоритм перестає бути оптимальним, так як відновити щільність по вибірці можна тільки з деякою погрішністю. Чим коротше вибірка, тим вище шанси підігнати розподіл під конкретні дані і зіткнутися з ефектом перенавчання.

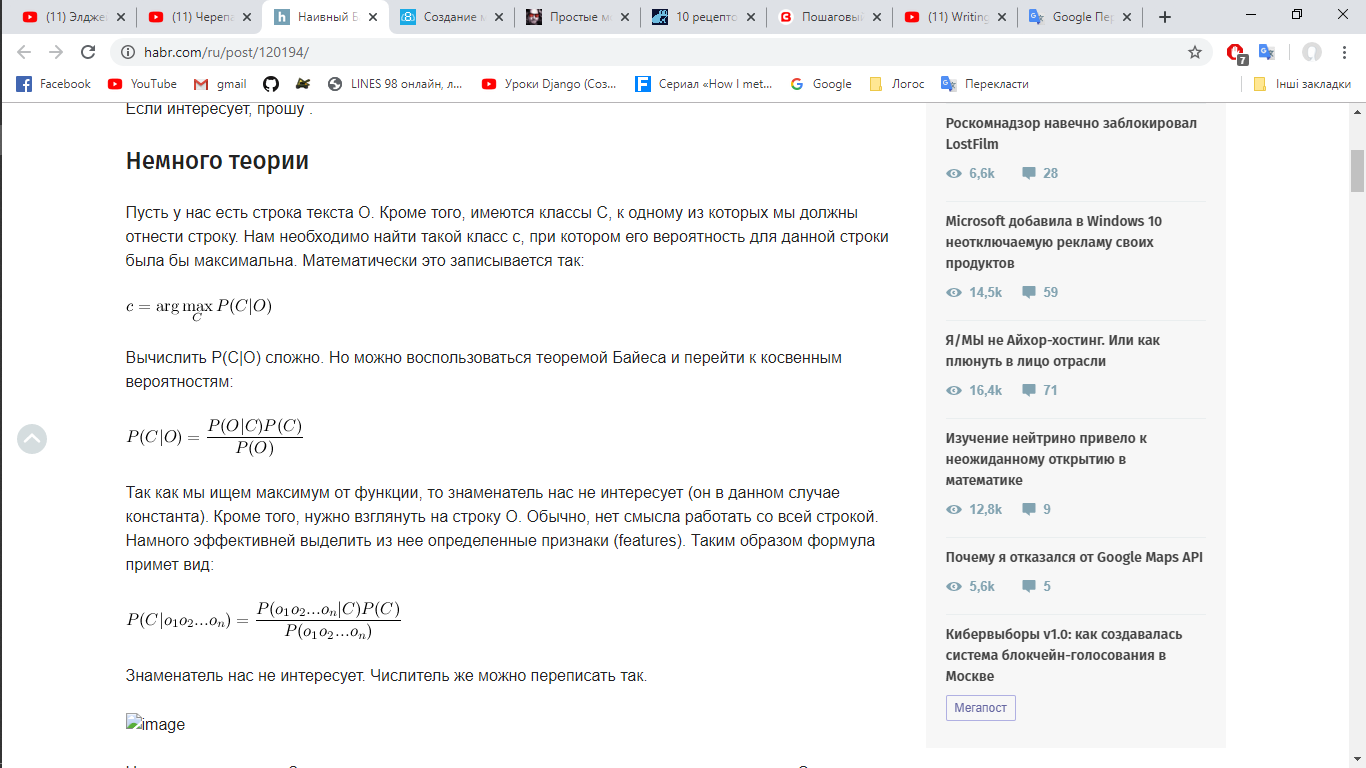
Байєсівський підхід до класифікації є одним з найстаріших, але до сих пір зберігає міцні позиції в теорії розпізнавання. Він лежить в основі багатьох досить вдалих алгоритмів класифікації.

**Алгоритм:**

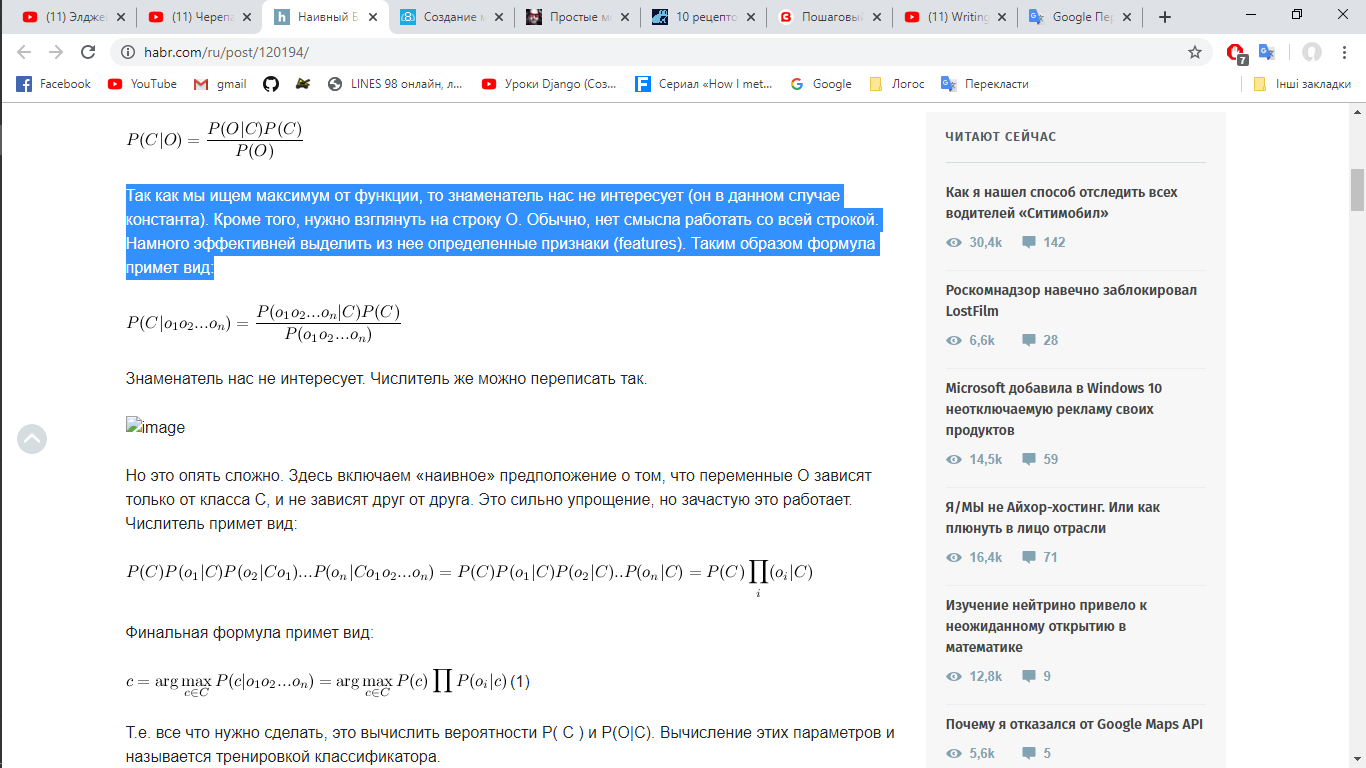
Нехай у нас є рядок тексту O. Крім того, є класи С, до одного з яких ми повинні віднести рядок. Нам необхідно знайти такий клас с, при якому його ймовірність для цього рядка була б максимальна. Математично це записується так:



Обчислити P (C | O) складно. Але можна скористатися теоремою Байєса і перейти до непрямих можливостей:

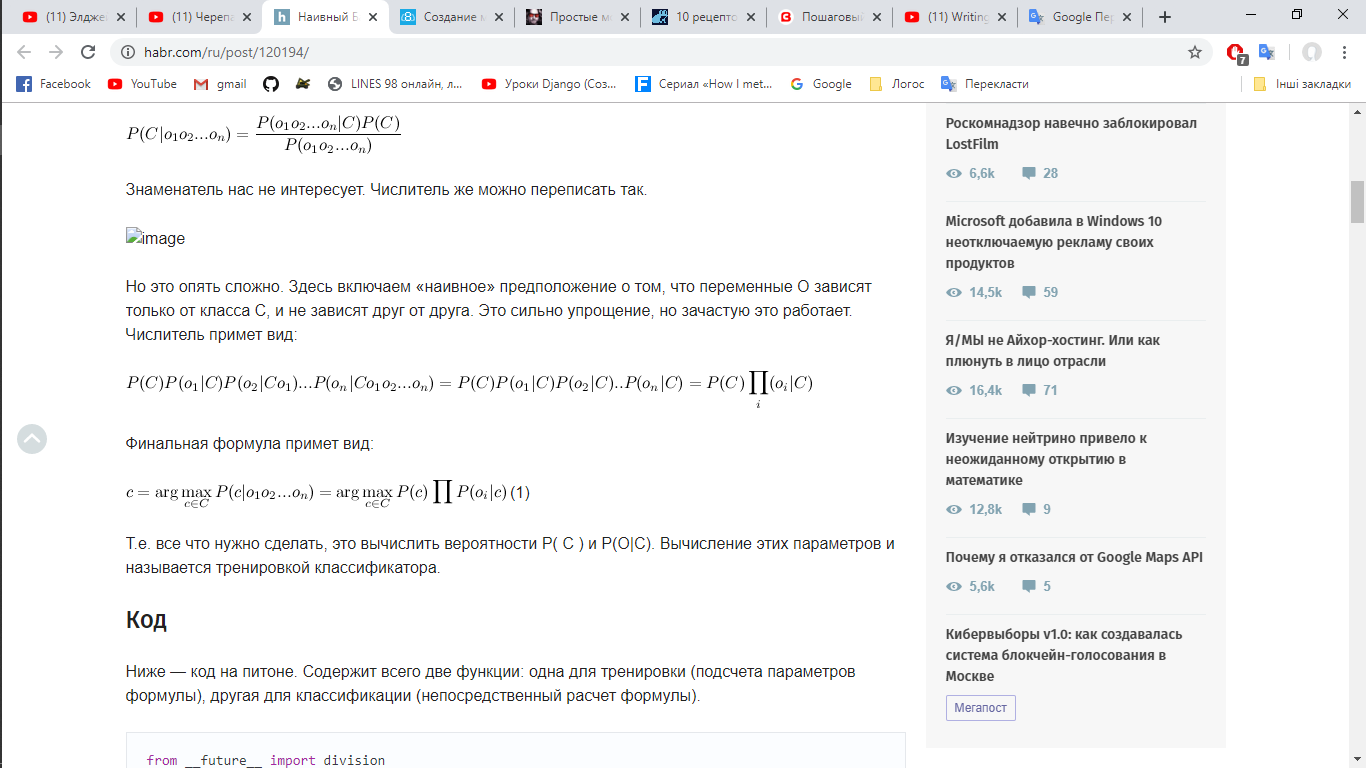


Так як ми шукаємо максимум від функції, то знаменник нас не цікавить (він в даному випадку константа). Крім того, потрібно поглянути на рядок O. Зазвичай, немає сенсу працювати з усією рядком. Набагато ефективніше виділити з неї певні ознаки (features). Таким чином формула набуде вигляду:



Все що потрібно зробити, це обчислити ймовірності P (C) і P (O | C). Обчислення цих параметрів і називається тренуванням класифікатора.

Фінальна формула має вигляд:



**Оцінка параметрів моделі:**

Всі параметри моделі можуть бути апроксимувати відносними частотами з набору даних навчання. Це оцінки максимальної правдоподібності ймовірностей. Безперервні властивості, як правило, оцінюються через нормальний розподіл. Як математичного очікування і дисперсії обчислюються статистики - середнє арифметичне і середнє квадратичне відхилення відповідно.

Якщо даний клас і значення властивості ніколи не зустрічаються разом в наборі навчання, тоді оцінка, заснована на можливостях, буде дорівнює нулю. Це проблема, так як при перемножуванні нульова оцінка призведе до втрати інформації про інших ймовірності. Тому бажано проводити невеликі поправки в усі оцінки ймовірностей так, щоб ніяка вірогідність не була строго дорівнює нулю.

**Проблема арифметичного переповнення:**

При досить великій довжині документа доведеться множити велика кількість дуже маленьких чисел. Для того щоб при цьому уникнути арифметичного переповнення знизу часто користуються властивістю логарифма твори ln(ab) = ln(a) + ln(b). Так як логарифм функція монотонна, її застосування до обох частин виразу змінить тільки його чисельне значення, але не параметри при яких досягається максимум. При цьому, логарифм від числа близького до нуля буде числом негативним, але в абсолютному значенні істотно більшим ніж вихідне число, що робить логарифмічні значення ймовірностей більш зручними для аналізу.

**Опис виконаного алгоритму:**

Можна бачити, що ідеальний байєсів класифікатор в якомусь сенсі є оптимальним. Його результат не може бути поліпшений, тому що у всіх випадках, коли можлива однозначна відповідь, він його дасть — а в тих випадках, коли відповідь неоднозначна, результат кількісно характеризує міру цієї неоднозначності.

Перевагою цього підходу є те, що вимоги до розміру вибірки скорочуються від експоненційних до лінійних. Недолік — те, що модель є точною лише у випадку, коли виконується припущення про незалежність. В іншому випадку, строго кажучи, обчислені ймовірності вже не є точними (і навіть більше того, їх сума може не дорівнювати одиниці, через що потрібно нормувати результат). Однак на практиці незначні відхилення від незалежності призводять лише до незначного зниження точності, і навіть у разі істотної залежності між змінними результат роботи класифікатора продовжує корелювати з істинною приналежністю образу до класів. При цьому переваги класифікатора (висока швидкість роботи, простота і масштабованість, помірні вимоги до пам'яті) часто переважують недоліки.

**ВИСНОВКИ**

Наївний байесовский класифікатор - простий імовірнісний класифікатор, заснований на застосуванні теореми Байеса зі строгими (наївними) припущеннями про незалежність.

Залежно від точної природи ймовірнісної моделі, наївні байєсовські класифікатори можуть навчатися дуже ефективно. У багатьох практичних додатках для оцінки параметрів для наївних Байесових моделей використовують метод максимальної правдоподібності; іншими словами, можна працювати з наївною байєсівської моделлю, не вірячи в Байєсову ймовірність і не використовуючи байєсовські методи.

Незважаючи на наївний вигляд і, безсумнівно, дуже спрощені умови, наївні байєсовські класифікатори часто працюють набагато краще в багатьох складних життєвих ситуаціях.

Перевагою наївного байєсівського класифікатора є мала кількість даних необхідних для навчання, оцінки параметрів і класифікації.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Writing Our First Classifier - Machine Learning Recipes #5

<https://www.youtube.com/watch?v=AoeEHqVSNOw&feature=youtu.be>

1. Классификатор на Пайтон

<https://shwanoff.ru/python-step-classifier/>

1. Наивный Байесовский классификатор

<https://habr.com/ru/post/120194/>

1. [СОЗДАНИЕ МАШИННО-ОБУЧАЕМОГО КЛАССИФИКАТОРА С ПОМОЩЬЮ SCIKIT-LEARN В PYTHON](https://www.8host.com/blog/sozdanie-mashinno-obuchaemogo-klassifikatora-s-pomoshhyu-scikit-learn-v-python/)

<https://www.8host.com/blog/sozdanie-mashinno-obuchaemogo-klassifikatora-s-pomoshhyu-scikit-learn-v-python/>

**ДОДАТКИ**

1. Код програми

from \_\_future\_\_ import division

from collections import defaultdict

from math import log

def read\_sample(file\_name):

sample = []

f = open(f"{file\_name}")

for line in f:

sample.append(line.strip().split(" "))

f.close()

return sample

def add\_to\_sample(file\_name):

f = open(f"{file\_name}", "a")

name, gender = input("Enter a name: ").capitalize(), input("Enter a gender (woman/man): ").lower()

print()

if gender in ["woman", "man"]:

f.write(f"{name} {gender}\n")

f.close()

def train(samples):

classes, freq = defaultdict(lambda:0), defaultdict(lambda:0)

for feats, label in samples:

classes[label] += 1

for feat in feats:

freq[label, feat] += 1

for label, feat in freq:

freq[label, feat] /= classes[label]

for c in classes:

classes[c] /= len(samples)

return classes, freq

def classify(classifier, feats):

classes, prob = classifier

return min(classes.keys(),

key = lambda cl: -log(classes[cl]) + \

sum(-log(prob.get((cl,feat), 10\*\*(-7))) for feat in feats))

def get\_features(sample): return (

'll: {sample[-1]}',

'fl: {sample[0]}',

'sl: {sample[1]}',

)

def get\_gender(file\_name, name):

samples = read\_sample(file\_name)

features = [(get\_features(feat), label) for feat, label in samples]

classifier = train(features)

return classify(classifier, get\_features(u'{}'.format(name.lower())))

def show\_info(name, gender):

print(f"""

Name : {name.capitalize()}

Gender: {gender.capitalize()}

""")

if \_\_name\_\_ == "\_\_main\_\_":

while True:

choice = input("""1 - teach

2 - check

Enter your choice: """)

print()

if choice == "2":

name = input("Enter a name: ")

gender = get\_gender(r'C:\Users\faryn\OneDrive\Рабочий стол\Лаби політех\Курсова\names.txt', name)

show\_info(name, gender)

elif choice == "1":

add\_to\_sample(r'C:\Users\faryn\OneDrive\Рабочий стол\Лаби політех\Курсова\names.txt')

else:

break