Софийски университет „Св. Кл. Охридски”

Факултет по математика и информатика

Курсов Проект

на тема:

„Класификация на текст

измежду повече от два класа”

Студент: **Константин Венциславов Нецов**, ФН: 26470,

спец. Информационни системи

Курс: „2-ри“, Учебна година: 2021/22

Преподаватели: **проф. Иван Койчев, Б. Величков**

==========

Декларация за липса плагиатство:

- Плагиатство е да използваш, идеи, мнение или работа на друг, като претендираш, че са твои. Това е форма на преписване.
- Тази курсова работа е моя, като всички изречения, илюстрации и програми от други хора са изрично цитирани.
- Тази курсова работа или нейна версия не са представени в друг университет или друга учебна институция.
- Разбирам, че ако се установи плагиатство в работата ми ще получа оценка “Слаб”.

8.7.22 г. Подпис на студента:

Съдържание

1 Увод

В областта на Машинното Самообучение задачата за класификация има широко приложение. Класификацията спада към обучението с учител от примери. В основната си форма тази задача представлява бинарна класификация или класифицирането на даден пример в една от две категории(класове). Пример за такъв вид класификация е дали да се отпусне заем на човек. Този проект разглежда задачата за класификация на текст измежду повече от два класа или *multiclass* класификация. Пример за такава задача е класификацията на документи. Данните за този проект представляват постове на теми съответстващи на класовете използвани за обучение. Постовете са събрани от форумите *stackoverflow.com* и *superuser.com* чрез *API* на *stackexchange*, част от която мрежа са тези два форума. Събраните данни ще бъдат обработени и почистени преди да бъдат подадени на съответните алгоритми. С цел да се намали броя на атрибутите и да се използват само най-информативните, ще се приложат методи за избор на атрибути и за намаляване на размерността. След това следва процеса по избиране на оптималните хиперпараметри за разглежданите модели. Накрая избраните модели ще бъдат обучени с оптималните хиперпараметри намерени в предната стъпка и сравнени използвайки няколко метрики, които да покажат колко добре се справят със задачата. В бъдеще най-добрия от разгледаните модели би могъл да се използва в задачи, като класификация на документи в един от класовете използвани при обучението.

2 Класификация на текст измежду повече от два класа

Алгоритмите за Машинно Самообучение не могат директно да работят с текст. Поради тази причина текста трябва да се представи в числова форма. Два от подходите за това представяне са *Word Embedding* и *Bag-of-words*. При подхода *Bag-of-words* се построява речник ,състоящ се от думите в корпуса от текстове. Всяка дума представя характеристика, а всеки ред е текст от корпуса. *Bag-of-words* има два варианта. При първия се брои колко пъти се среща съответната дума в текста. Втория наречен *Term Frequency Inverse Document Frequency (TF-IDF)* е статистическа мярка за това колко е релевантна дума за текст в корпус от текстове. На Фигура 1, Фигура 2 и Фигура 3 е показано как се изчислява тази мярка.

$$TFIDF = TF(w, d) * IDF(w)$$

Фигура 1

$$TF(i,j) = \frac{\text{Term i frequency in document j}}{\text{Total words in document j}}$$

Фигура 2

$$IDF(i) = \log_2 \left(\frac{\text{Total documents}}{\text{documents with term i}} \right)$$

Фигура 3

При *Bag-of-words* не се взима предвид позицията на думата в текста. Тъй като при този метод големият речник генерира голямо пространсто върху което трябва алгоритмите да работят, може да се използват техники за избор на атрибути или за намаляване на размерността на пространството. *Principal Component Analysis (PCA)* е една техника за намаляване на размерността на пространството. При нея атрибутите се проектират в пространство с по-малка размерност, като това пространство съдържа голяма част от информацията от оригиналното пространство. За избор на атрибути може да се ползва χ_2 статистика. При нея се проверява нулевата хипотеза, че атрибута и съответния клас са независими. Ако нулевата хипотеза се отхвърли, то атрибута и класа са зависими и този атрибут би бил полезен в обучението на моделите. Другия метод за представяне на текст в числова форма е *Word Embedding*. При него думите се представят в пространство с по-ниска размерност. Той позволява векторите представящи думи с близко значение да бъдат близки в това пространство. Повечето от алгоритмите поддържат само бинарна класификация, като например *Support Vector Machine*. За да може даден алгоритъм да работи с повече от два класа може да се използват техниките *One-Vs-Rest(OvR)* и *One-Vs-One(OvO)*. При *OvR* се конструират толкова бинарни класификатора колкото е броя на класовете. При всеки от тези бинарни класификатори, позитивния клас е този за който се строй класификатора , а всички останали класове се разглеждат като негативния. При *OvO за* **N** класа се създават **N*(N-1)/2** класификатора**.** За всяка двойка класове, от трениращото множество за трениране се взимат само примери от тези два класа. Друг подход за класификация на текст измежду повече от два класа са дълбоките неврони мрежи. При тях на последния слой

броя на възлите е равен на броя на класовете. Активиращата функция за всеки от тези възли е *softmax.*

3 Проектиране

Проекта е съставен от две части. Първата се занимава със събирането на данните и привеждането им в таблична форма, нужна за последваща работа с тях. Тези функции са реализирани чрез Python скрипт. Този скрипт изпълнява следните функции:

- Изпращане на завки към api.stackexchange.com.
- Обработка на върнатите данни от api.stackexchange.com.
- Обединяване на обработените данни в табличен формат подходящ за анализ и обуванате на модели.

Втората част е реализирана в Jupyter Notebook. В него е представен процеса по анализ, обработка, обучението на моделите и тяхната оценка.

4 Реализация, тестване/експерименти

4.1 Използвани технологии, платформи и библиотеки

За реализирането на проекта е използван езика за програмиране Python и следните библиотеки:

- Numpy
- Pandas
- Sklearn
- Matplotlib
- Beautiful Soup

4.2 Реализация

За заявки към api.stackexchange.com е използвана Python библиотеката requests. Отгорите от този *API* са в *html* формат и са обработени с библиотеката beautifulSoup. Обработените *html* файлове са запазени като *json* файлове. За всеки от класовете използвани за предсказване има такива файлове, съдържащи постовете които ще се използват за обучение. Крайното множество от данни е в

резултат от обединяването на тези *json* файлове. След като данните са събрани и са трансформирани в табличен вид следва предварителната им обработка. Тя се състой в разбиването на текста на отделни думи(*token*-и), премахване на стоп думите, премахване на нерелевантни думи и лематизация. Направен е базов анализ на данните, както и визуализации като показване на разпределението на примерите по класове. За представяне на текста в числов вид е използван *TF-IDF*. Данните се разделят 80% за обучение и 20% за тестване. Приложени са техниката за избор на атрибути χ^2 статистика и *Singular Value Decomposition* за намаляване на размерността на пространството с което ще се работи. Разгледани са следните алгоритми: *Support Vector Machine, Logistic Regression, Naïve Bayes*, както и алгоритъм предсказваш най-често срещания клас в данните, като той служи за отправна точка за предсказващите способности на останалите модели. Върху обучаващите данни се прилага процедурата за определяне на оптималните хиперпараметри на алгоритмите изброени по-горе - *RandomizedSearchCV*. Посочените модели се обучават, като се използват оптималните хиперпараметри намерени в предната стъпка. Накрая се прави съпоставка върху различни метрики измерващи предсказващите способности на моделите.

5 Заключение

Поради малкото обучаващи примери е възможно разгледаните модели да дават завишени резултати върху трениращите данни и да не генерализират добре върху нови примери. Това е познато като пренагаждане или *overfitting.*

Възможността за пренагаждане се увеличава в случаите, когато атрибутите са повече от обучаващите примери. За справяне с този проблем са използвани χ^2 статистика за избор на атрибутите и *Singular Value Decomposition* за намаляване на размерността на пространството, което ще се използва за обучение на моделите. Друга възможност за преодоляване на този проблем е събирането на повече данни. Едно от бъдещите подобрения може да е включването на още класове. За представянето на текста в числов вид може да се използва и *Word Embedding.* Този подход използва невронни мрежи. Също така за решаване на проблема може да се изпробва подхода с дълбоки невронни мрежи с различни архитектури. Те могат да се комбинират с метода *Word Embedding,* което потенциално би довело до по-добри резултати при класификация, особено при използването на голям брой обучаващи данни.

6 Използвана литература

1. **https://medium.com/pythoneers/nlp-word2vec-with-python-example-3713e3157809**
2. **https://towardsdatascience.com/introduction-to-text-representations-for-language-processing-part-1-dc6e8068b8a4**
3. **https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/**
4. **https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b**