به نام خداوند بخشنده و مهربان

درس داده کاوی

تمرین سوم

استاد درس: دکتر ناظرفرد

نام دانشجو:

روزبه قاسمی ۹۵۳۱۴۲۴

اردیبهشت ۱۳۹۹

سوال اول

تعریف Gradient Tree Boosting:

درختان تقویت کننده گرادیان در حال حاضـر بهترین تکنیک های سـاخت مدل های پیش بینی کننده از داده های سـاختاری هسـتند. بر خلاف مدل ها برای تجزیه و تحلیل تصـاویر (برای این که می خواهید از یک مدل یادگیری عمیق استفاده کنید) ، مشکلات داده های ساخت یافته می تواند با درختان تصمیم گیری زیادی حل شـود. بهترین بسـته های تصـمیم گیری درخت می توانند به مجموعه داده های بزرگ با پیچیدگی زمانی زیر خطی آموزش دهند ، به موازات بسـیاری از cpus، در بین خوشـه ای از رایانه ها توزیع شـده و می توانند اکثر اطلاعات را از یک مجموعه داده در زیر یک دقیقه استخراج کنند. این الگوریتم شبیه به AdaBoost است که هر دو از یک گروه از درخت‌های تصمیم‌گیری برای پیش‌بینی یک برچسب هدف استفاده می‌کنند. با این حال، برخلاف AdaBoost، درختان تقویت کننده گرادیان عمق بیشتری نسبت به ۱ دارند.

تعریف Random Forest:

جنگل تصادفی مانند نام آن، شامل تعداد زیادی درخت تصمیم‌گیری فردی است که به عنوان یک گروه کار می‌کنند. هر درخت منفرد در جنگل تصادفی یک پیش‌بینی کلاس را نشان می‌دهد و کلاس با بیش‌ترین آرا پیش‌بینی مدل ما می‌شود (شکل زیر را ببینید). مفهوم بنیادی در پشت جنگل تصادفی ساده اما قدرتمند است - حکمت جمعیت. در علوم داده‌ها، دلیل اینکه مدل جنگل تصادفی به خوبی کار می‌کند این است: تعداد زیادی از مدل‌های نسبتا ناهمبسته (درختان)که به عنوان یک کمیته عمل می‌کنند، از هر کدام از مدل‌های مجزای انفرادی پیشی خواهند گرفت. رابطه پایین بین مدل‌ها کلیدی است. درست مانند نحوه سرمایه‌گذاری با همبستگی پایین (مانند سهام و اوراق‌قرضه)برای تشکیل یک پورتفولیو که بزرگ‌تر از مجموع قطعات آن است، مدل‌های ناهمبسته می‌توانند پیش‌بینی‌های گروهی را تولید کنند که دقیق‌تر از هر کدام از پیش‌بینی‌های فردی هستند. دلیل این افکت شگفت‌انگیز این است که درخت‌ها یکدیگر را از خطاهای فردی خود محافظت می‌کنند (تا زمانی که همیشه در مسیر یک‌سان خطا نمی‌کنند). در حالی که ممکن است برخی درختان اشتباه باشند، بسیاری از درختان دیگر درست خواهند بود، بنابراین به عنوان گروهی از درختان قادر به حرکت در جهت درست هستند. پس پیش‌نیازها برای جنگل تصادفی برای اجرای خوب این است:

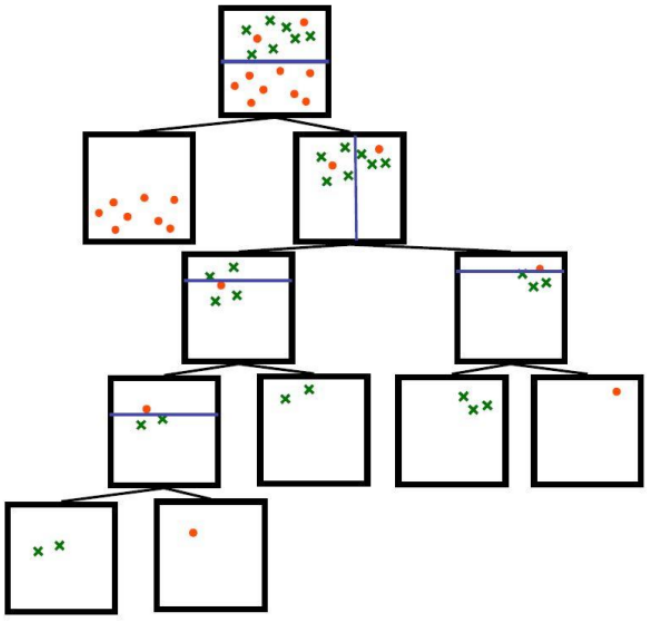
۱) در اینجا نیاز به سیگنال واقعی در ویژگی‌های ما وجود دارد، بنابراین مدل‌هایی که با استفاده از این ویژگی‌ها ساخته شده‌اند، بهتر از حدس زدن تصادفی عمل می‌کنند.
۲) پیش‌بینی‌ها (و بنابراین خطاهای انجام‌شده) توسط درختان منفرد باید همبستگی پایینی با یکدیگر داشته باشند.

دو تفاوت اصلی آنها عبارتند از:

۱) **چگونه درختان ساخته می‌شوند:** جنگل‌های تصادفی هر درخت را مستقل می‌سازد در حالی که افزایش گرادیان یک درخت را در یک زمان می‌سازد. این مدل افزایشی (ensemble) در یک روش به جلو عمل می‌کند، و یک یادگیرنده ضعیف را معرفی می‌کند تا کمبودهای موجود ضعیف موجود را بهبود بخشد.
۲) **نتایج ترکیب نتایج:** جنگل‌های تصادفی نتایج را در پایان فرآیند ترکیب می‌کنند (به طور متوسط یا "قواعد اکثریت")در حالی که افزایش گرادیان ترکیب نتایج را در طول مسیر ترکیب می‌کند.

اگر به دقت پارامترها را تنظیم کنید، افزایش شیب می‌تواند منجر به عملکرد بهتری نسبت به جنگل‌های تصادفی شود. با این حال، اگر نویز زیادی داشته باشید، افزایش گرادیان ممکن است گزینه مناسبی نباشد چون می‌تواند منجر به Overfitting شود. آن‌ها همچنین تمایل دارند سخت‌تر از جنگل‌های تصادفی باشند. جنگل‌های تصادفی و شیب هر کدام در مناطق مختلف پیشرفت می‌کنند. جنگل‌های تصادفی به خوبی برای شناسایی اشیا چند طبقه و بیوانفورماتیک به خوبی عمل می‌کنند که این امر منجر به نویز آماری زیادی می‌شود. عملکرد Boosting وقتی خوب عمل می‌کند که داده‌های نامتوازن مثل ارزیابی ریسک زمان واقعی داشته باشید.

سوال دوم



از نظر من دو ایراد در درخت بالا موجود است:

سمت راست درخت می‌بایستی حرس شود زیرا باعث شده است واریانس بالا برود. این واریانس بالا باعث می‌شود که به صورت خطی جدا داده‌ها را جدا کند که این تقسیم‌بندی درست نیست!

سوال سوم

طبق خواسته سوال، زمانی معیار Accuracy ، معیاری مناسب نخواهد بود که نسبت داده‌های ما با هم هم‌خوانی نداشته باشد. به عبارتی ساده تر فرض کنید ما هدف‌مان پردازش زبان طبیعی[1] و در حوضه تحلیل احساسات[2] می‌باشد. اگر تعداد داده‌های مثبت با تعداد داده‌های منفی هم‌خوانی نداشته باشد، در نتیجه فارغ از اینکه از چه مدلی برای آموزش استفاده کرده‌ایم نتیجه به دست آمده بایاس خواهد شد. به این صورت که اگر داده‌های مثبت فرضاً ۹۰ درصد و داده‌های منفی ۱۰ درصد از مجموعه داده‌آموزشی ما باشند، در زمان تست هر جمله ای که به مدل می‌دهیم، آنرا مثبت پیش‌بینی[3] می‌کند. در نتیجه معیار Accuracy در اینجا مناسب نخواهد بود.

حال برای این مشکل می‌توانیم از سه معیار دیگر استفاده کنیم:

۱) دقت[4]

معیار دقت نسبت تعداد پیش‌بینی‌های صحیح انجام شده برای نمونه‌های یک کلاس خاص، به تعداد کل پیش‌بینی‌ها برای نمونه‌های همان کلاس خاص را (این تعداد، مجموع تمامی پیش‌بینی‌های صحیح و پیش‌بینی‌های نادرست را شامل می‌شود) ارزیابی می‌کند. مقدار بالا برای معیار دقت، بیانگر تعداد کم داده‌هایی است که به اشتباه، در کلاس خاص دسته‌بندی شده‌اند. شایان توجه است که معیار دقت، فقط برای مواردی ارزیابی می‌شود که در آن‌ها، مدل دسته‌بندی تعلق یک نمونه به یک کلاس خاص را پیش‌بینی کرده باشد. این امتیاز از رابطه زیر بدست می‌آید:

$$Precision = \frac{TP}{TP + FP}$$

[1] Natural Language Processing
[2] Sentiment Analysis
[3] Prediction
[4] Precision

۲) یادآوری

معیار یادآوری، بیان کننده نسبت تعداد داده‌های متنی درست دسته‌بندی شده در یک کلاس خاص، به تعداد کل داده‌هایی است که باید در همان کلاس خاص دسته‌بندی شوند. مقدار بالا برای معیار یادآوری، بیانگر تعداد کم داده‌هایی است که به اشتباه، در آن کلاس خاص دسته‌بندی نشده‌اند. استفاده از این معیار، به تنهایی، برای ارزیابی عملکرد سیستم درست نیست و باید در کنار معیار دقت مورد استفاده قرار بگیرد. زیرا، به راحتی می‌شود مدل‌های دسته‌بندی متنی طراحی کرد که یادآوری بالایی داشته باشند و این لزوما به معنای دقت بالا نیست. این معیار از رابطه زیر بدست می‌آید:

$$Recall = \frac{TP}{TP + FN}$$

۳) F1 - Score

این معیار، پارامترهای دقت و یادآوری با هم ترکیب می‌کند تا مشخص شود یک مدل دسته‌بند تا چه حد عملکرد خوبی از خود نشان می‌دهد. این امتیاز از رابطه زیر بدست می‌آید:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

سوال چهارم

فرض کنید ۲۵ طبقه‌بندی کننده پایه وجود دارد و هر طبقه بند دارای نرخ خطا می‌باشد که به معنی آن است که طبقه‌بندی کننده‌ها مستقل هستند یعنی احتمال اینکه یک طبقه‌بندی کننده به اشتباه تبدیل شود به این که آیا طبقه‌بندی کننده‌های دیگر اشتباه کرده‌اند بستگی ندارد. احتمال اینکه طبقه بند دسته‌بندی کننده یک پیش‌بینی غلط ایجاد کند، اگر اکثر طبقه‌بندی کننده‌ها یک پیش‌بینی غلط داشته باشند، احتمال اینکه ۱۳ یا بیشتر طبقه‌بندی کننده خطا باشد، یک پیش‌بینی غلط ایجاد می‌کند.

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} \approx 0.06 \ll \varepsilon$$

سوال پنجم

الف) برای حل این سوال باید هر بار یکی از صفات در ریشه قرار بگیرد و آن صفتی که بیشترین Gain را داشته باشد، باید در ریشه قرار بگیرد و در نتیجه درخت به دست آمده بر اساس همان ریشه است.

Class positive : Edible = "Edible"

Class Negative : Edible = "Poisonous"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)$$

$$= 0/940$$

- اگر صفت Habitat در ریشه باشد:

$$Info_{Habitat}(D) = \frac{5}{14} \times \left(-\frac{3}{5}\log_2\left(\frac{2}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right) \Rightarrow Woods$$

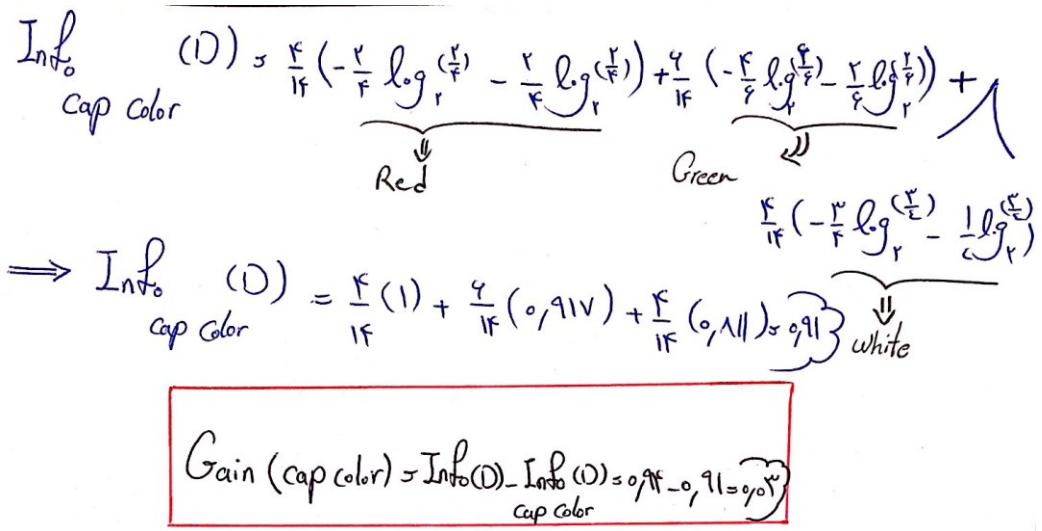
$$+ \frac{4}{14} \times \left(-\frac{4}{4}\log_2\left(\frac{4}{4}\right)\right) \Rightarrow Grasses$$

$$+ \frac{5}{14} \times \left(-\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right) \Rightarrow Leaves$$

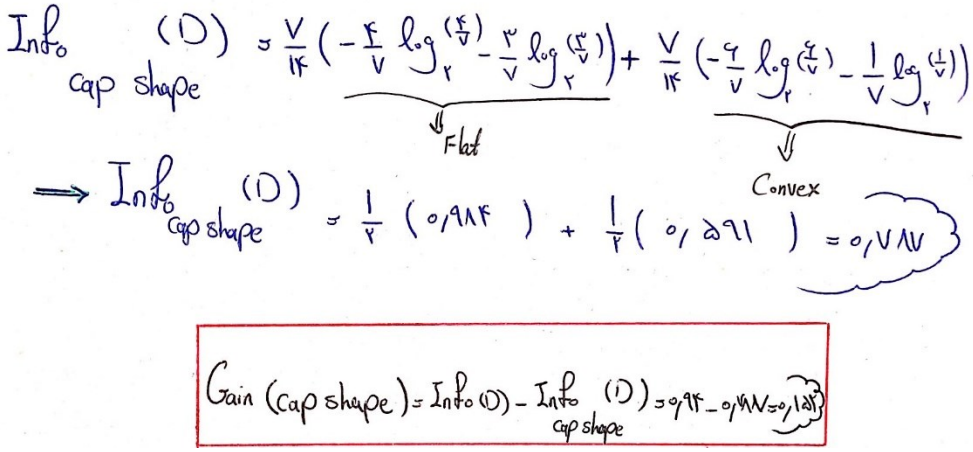
$$\Rightarrow Info_{Habitat}(D) = 0/694$$

$$Gain(Habitat) = Info(D) - Info_{Habitat}(D) = 0/246$$

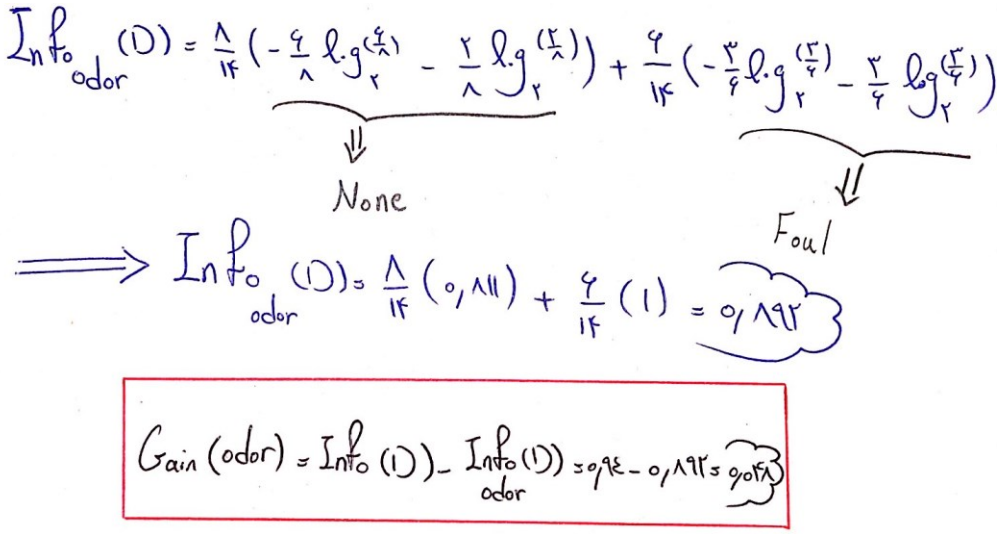
- اگر صفت Cap Color در ریشه باشد:



- اگر صفت Cap Shape در ریشه باشد:



- اگر صفت Odor در ریشه باشد:

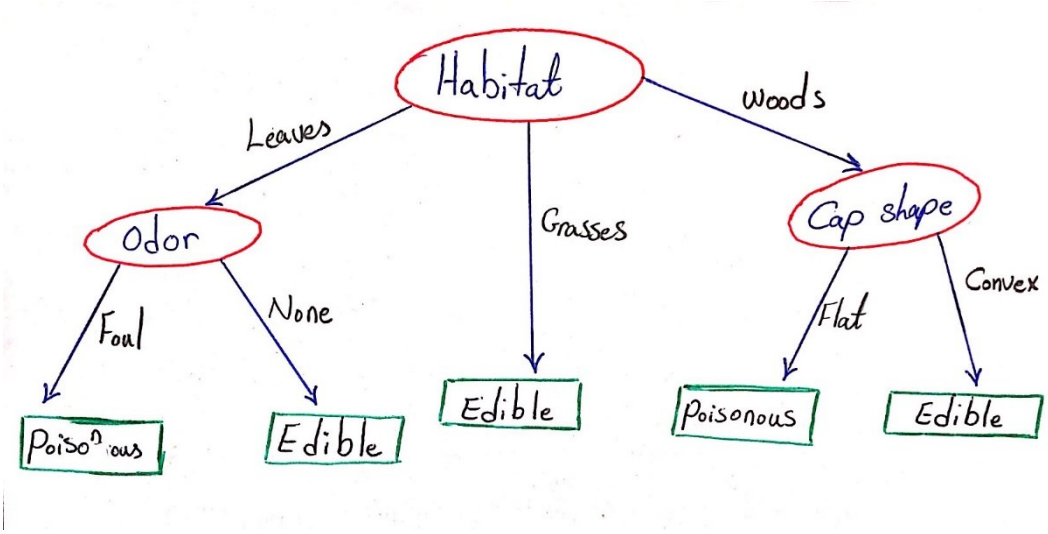


بر اساس نتایج بدست‌آمده جدول آنرا رسم می‌کنیم:

	Habitat	Cap Color	Cap Shape	Odor
Gain	**0.246**	**0.03**	**0.152**	**0.048**

بر اساس جدول بالا می‌گیریم که Habitat باید در ریشه قرار بگیرد.

ب) حال درخت را رسم می‌کنیم:



سوال ششم

الف)

$$P(y \mid a,b,c) = \frac{P(a \mid y)P(b \mid y)P(c \mid y)P(y)}{\sum_{y'} P(a \mid y')P(b \mid y')P(c \mid y')P(y')}$$

ب)

اگر y- را به معنای Bad و y را به معنای OK بگیریم داریم:

$$P(y \mid -a, b, -c) = \frac{2}{3} \times \frac{1}{3} \times \frac{2}{3} \times \frac{2}{3} = \frac{1}{18}$$

$$P(-y \mid -a, b, -c) = \frac{3}{5} \times \frac{2}{3} \times \frac{2}{5} \times \frac{2}{5} = \frac{4}{25}$$

$$\text{Posterior probability} = \frac{4 \times 18}{4 \times 18 + 25} = \frac{72}{97}$$

گزارش بخش پیاده‌سازی

پیاده سازی اول ؛ الگوریتم Decision Tree

درخت تصمیم یکی از قوی‌ترین و پرکاربردترین الگوریتم‌های داده‌کاوی است که برای کاوش در داده‌ها و کشف دانش کاربرد دارد. این الگوریتم داده‌ها را به مجموعه‌های مشخصی تقسیم می‌کند. هر مجموعه شامل چندین زیر مجموعه از داده‌های کم و بیش همگن که دارای ویژگی‌های قابل پیش بینی هستند تقسیم می‌شود. در این پیاده‌سازی نیز ما با داده‌های پزشکی کار داریم. در ابتدا می‌بایست سن افراد را بازه بندی کنیم که طبق بازه‌بندی داده شده کد آن به شکل زیر است:

```
def change_ages(age):
    if(0<= age <= 10):
        return 0
    elif ((11<= age <= 20)):
        return 1
    elif ((21<= age <= 30)):
        return 2
    elif ((31<= age <= 40)):
        return 3
    elif ((41<= age <= 50)):
        return 4
    elif ((51<= age <= 60)):
        return 5
    elif ((61<= age <= 70)):
        return 6
    elif ((71<= age <= 80)):
        return 7
data['age'] = data.age.apply(change_ages)
```

سپس داده‌های غیر عددی را از طریق کد زیر تبدیل به داده های عددی کنیم:

```
data.sex = data.sex.astype('category')
data.cp = data.cp.astype('category')
data.fbs = data.fbs.astype('category')
data.exang = data.exang.astype('category')
data.thal = data.thal.astype('category')
data.target = data.target.astype('category')

dummies = pd.get_dummies(data = data , drop_first= True)
dummies.head()
```

سپس Head داده‌ها را برای صحت عملکرد کدهای قبلی مشاهده می‌کنیم:

	age	trestbps	chol	restecg	thalach	oldpeak	slope	ca	sex_male	cp_none	cp_severe	cp_weak	fbs_True	exang_yes	thal_fixed_defect	thal_normal	target_yes
0	6	145	233	0	150	2.3	0	0	1	0	1	0	1	0	0	1	1
1	3	130	250	1	187	3.5	0	0	1	0	0	0	0	0	1	0	1
2	4	130	204	0	172	1.4	2	0	0	0	0	1	0	0	1	0	1
3	5	120	236	1	178	0.8	2	0	1	0	0	1	0	0	1	0	1
4	5	120	354	1	163	0.6	2	0	0	1	0	0	0	1	1	0	1

سپس طبق درخواست مسئله باید داده‌ها را به داده‌های Train و Test تبدیل کنیم، ۸۰ درصد داده آموزشی و ۲۰ درصد داده تست:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=101)
```

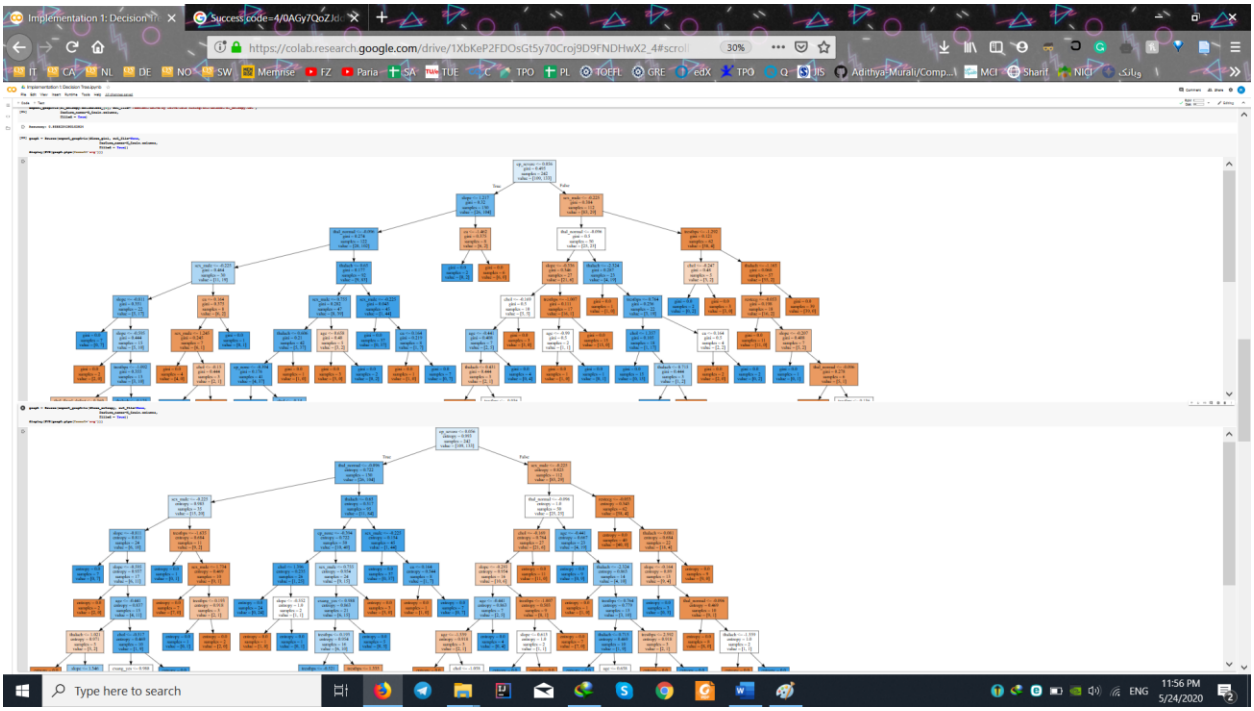
حال به آموزش مدل می‌پردازیم:

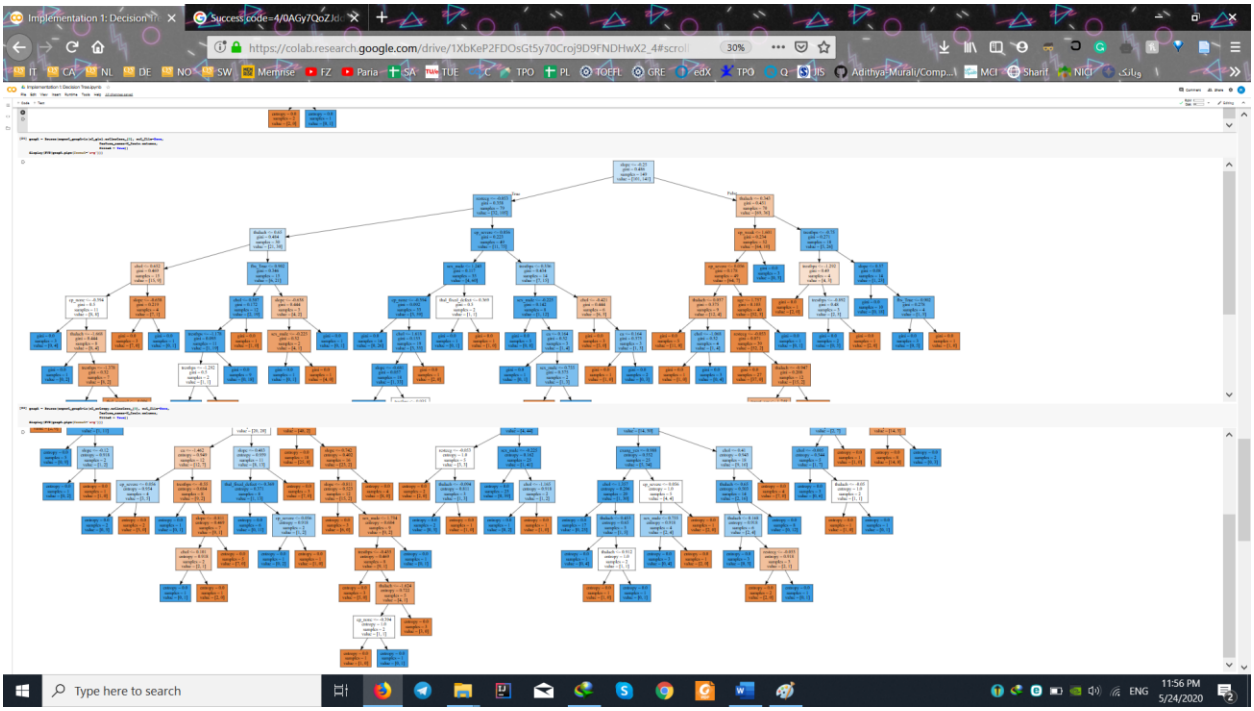
```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()

dtree.fit(X_train,y_train)

y_pred = dtree.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix,classification_report

print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.71      0.77        31
           1       0.74      0.87      0.80        30

    accuracy                           0.79        61
   macro avg       0.79      0.79      0.79        61
weighted avg       0.80      0.79      0.79        61
```

حال مقادیر مختلف برای معیار را امتحان کنیم و درختان زیر بدست می‌آید:





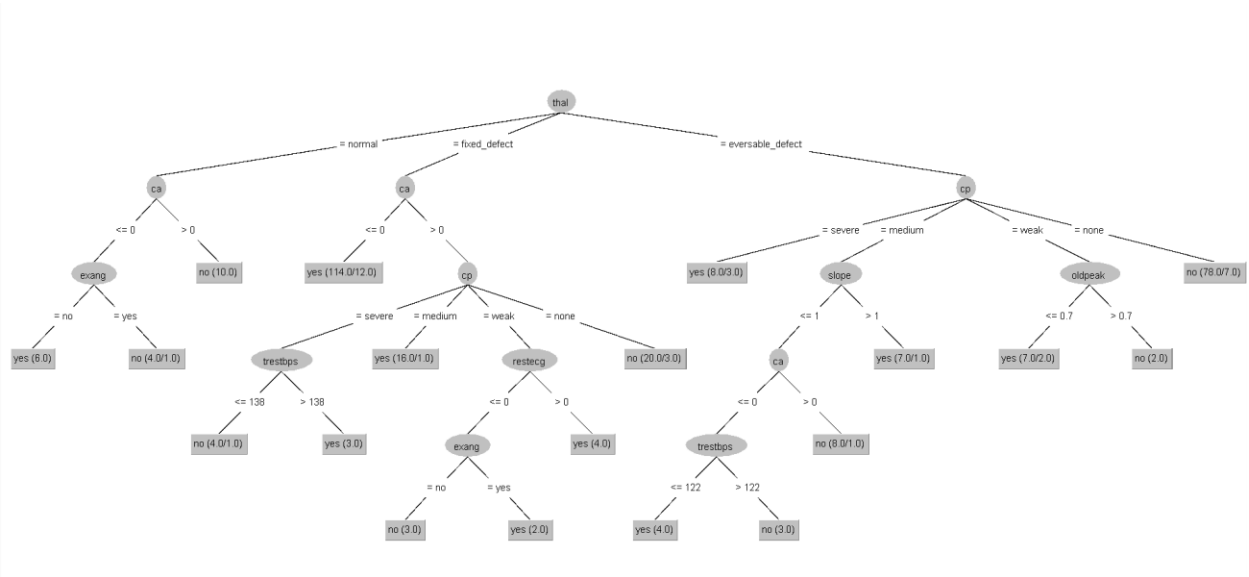
```
[32] from sklearn.metrics import confusion_matrix,classification_report

     print(classification_report(y_test,y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.86      0.86      0.86        29
               1       0.88      0.88      0.88        32

        accuracy                           0.87        61
       macro avg       0.87      0.87      0.87        61
    weighted avg       0.87      0.87      0.87        61
```

- چندبار تلاش کردم بتونم درخت‌ها رو به صورت عکس دانلود کنم ولی نشد به همین خاطر اسکرین شات رو گذاشتم، با عرض پوزش!

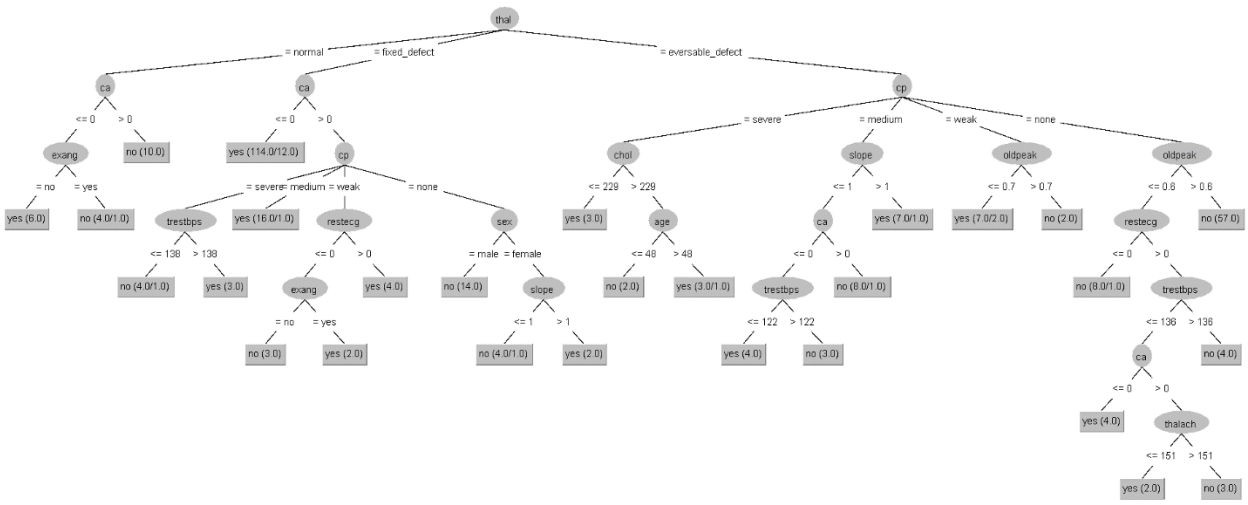
پیاده سازی دوم ؛ Decision Tree with Weka

Weka دارای چندین تابع برای اجرای طیف وسیعی از الگوریتم‌های یادگیری ماشین از رگرسیون خطی تا شبکه عصبی است. این به شما این امکان را می‌دهد که پیچیده‌ترین الگوریتم روی مجموعه داده خود را تنها با کلیک روی یک دکمه قرار دهیم. علاوه بر این، Weka از دسترسی به برخی از رایج‌ترین الگوریتم های کتابخانه یادگیری ماشین پایتون و R پشتیبانی می‌کند. با Weka شما می‌توانید داده‌ها را تجزیه کنیم، داده‌ها را دسته‌بندی کنیم و حتی داده‌ها را تجسم کنیم! این کار را می‌توانید در فرمت های مختلف فایل‌های داده مانند ARFF، CSV، C4.5، و JSON. Weka حتی به شما اجازه می‌دهد تا فیلتر را به مجموعه dataset اضافه کنیم که از طریق آن می‌توانید داده‌های خود را نرمال کنیم، آن را استاندارد کنیم. برای پیاده‌سازی این قسمت از کدی که ضمیمه شده است، فایل csv را به فرمت ARFF تبدیل کردم سپس آنرا از طریق راهنمایی که در صورت تمرین قرار گرفته بود وارد برنامه Weka کردم و نتایج زیر بدست آمد.



درخت بدست‌آمده با پارامتر CF = 0.05

```
Number of Leaves  :     2

Size of the tree :      3


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         302               99.67   %
Incorrectly Classified Instances         1                0.33   %
Kappa statistic                          0.9934
Mean absolute error                      0.0033
Root mean squared error                  0.0574
Relative absolute error                  0.6653 %
Root relative squared error             11.5347 %
Total Number of Instances              303

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.994    0.000    1.000      0.994    0.997      0.993    0.997     0.997     yes
               1.000    0.006    0.993      1.000    0.996      0.993    0.997     0.993     no
Weighted Avg.  0.997    0.003    0.997      0.997    0.997      0.993    0.997     0.995

=== Confusion Matrix ===

   a   b   <-- classified as
 164   1 |   a = yes
   0 138 |   b = no
```

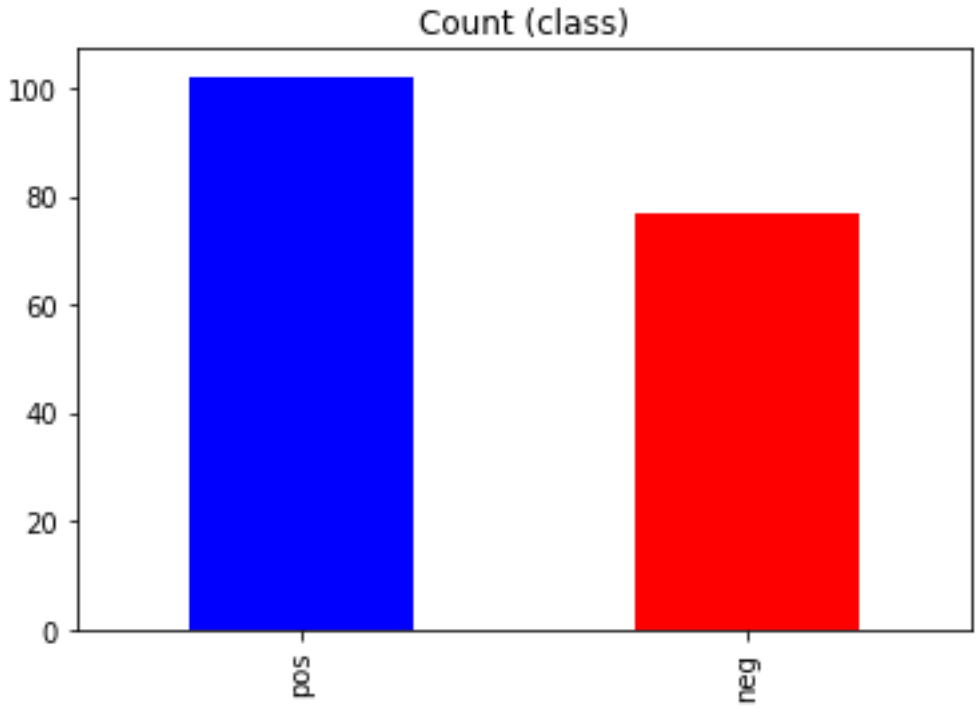


درخت بدست‌آمده با پارامتر CF = 0.35

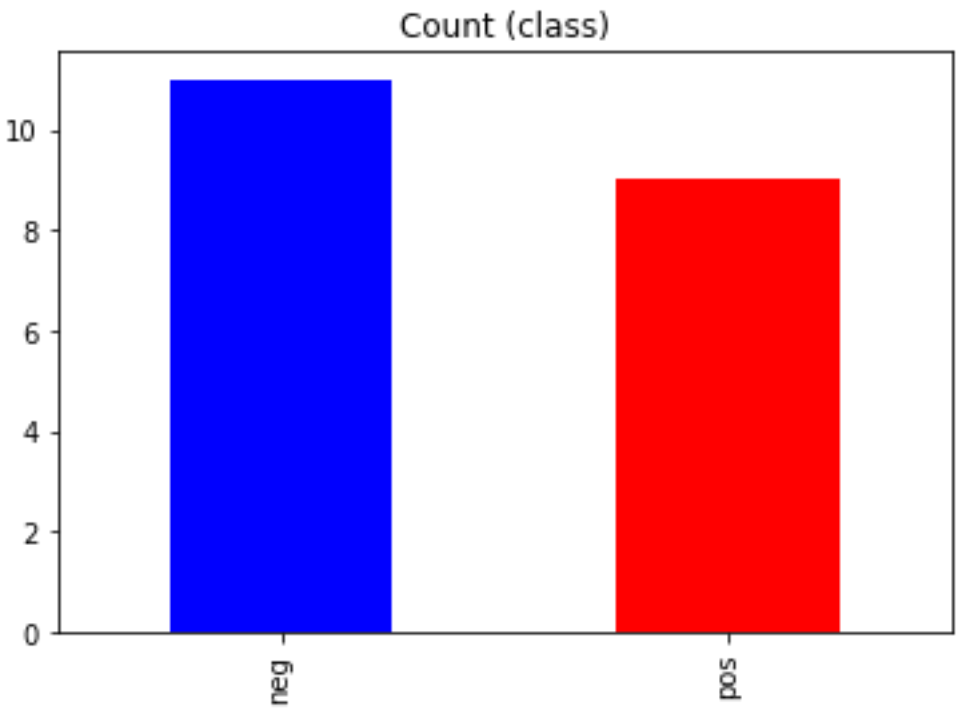
پیاده سازی سوم ؛ Naïve Bayes Text Classification

در این پروژه multinomial Naive بیز (sklearn و همچنین multinomial Naive بیز)برای طبقه‌بندی متن با استفاده از پایتون ۳ استفاده شده‌است. دیتاست مورد استفاده مسئله را ابتدا تحلیل و مقادیر مثبت و منفی را بدست آوردم.

در ابتدا تحلیل مقادیر مثبت و منفی داده‌های آموزش و در ادامه تحلیل داده‌های تست را نشان می‌دهیم.



Negative: 102
Positive: 77
Proportion: 1.32 : 1



Negative: 11
Positive: 9
Proportion: 1.22 : 1

سپس با استفاده از stop words ها، داده‌ها را پیش پردازش و توکنایز کردیم.

```
train_size = train_data.size
test_size = test_data.size

print("Amount of Train Data is :" ,train_size)
print("Amount of Test Data is :" ,test_size)
```

```
Amount of Train Data is : 358
Amount of Test Data is : 40
```

```
# Load Stop Words
stop_words = pd.read_csv('/content/drive/My Drive/Data Mining/HW3/dataset/sw.txt', header = None)
```

```
import nltk
import string
import re
```

```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

def text_processing(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text)
    translator = str.maketrans('', '', string.punctuation)
    text = text.translate(translator)
    text = " ".join(text.split())
    word_tokens = word_tokenize(text)
    text = [word for word in word_tokens if word not in stop_words]
    # provide context i.e. part-of-speech
    text = [lemmatizer.lemmatize(word, pos ='v') for word in word_tokens]
    return text
data = []
for i in range(179):
    data.append(text_processing(train_Review[i]))
```

حال برای درک بهتر یک جمله را قبل و بعدش را مقایسه می‌کنیم، به طور مثال جمله دوم با ایندکس ۱:

```
train_Review[1]
```

```
" The best soundtrack ever to anything.: I'm reading a lot of reviews saying that this is the best 'game soundtrack' and I figured
```

```
data[1]
```

```
['the',
 'best',
 'soundtrack',
 'ever',
 'to',
 'anything',
 'im',
 'read',
 'a',
 'lot',
 'of',
 'review',
 'say',
 'that',
 'this',
 'be',
 'the',
 'best',
 'game',
 'soundtrack',
 'and',
 'i',
 'figure',
 'that',
 'id',
 'write',
 'a',
```

سپس به سراغ پیاده سازی Naïve Bayes می‌رویم، برای جلوگیری از طولانی شدن گزارش صرفاً نتایج ذکر می‌شود:

```
Our score on testing data : 0.861
Classification report for testing data :-
                          precision    recall  f1-score   support

             alt.atheism       0.73      0.80      0.77       233
           comp.graphics       0.78      0.79      0.79       253
 comp.os.ms-windows.misc       0.83      0.82      0.83       249
comp.sys.ibm.pc.hardware       0.81      0.88      0.84       240
   comp.sys.mac.hardware       0.85      0.92      0.89       236
          comp.windows.x       0.93      0.83      0.88       240
            misc.forsale       0.81      0.87      0.84       261
               rec.autos       0.90      0.92      0.91       269
         rec.motorcycles       0.90      0.97      0.93       284
      rec.sport.baseball       0.99      0.97      0.98       248
        rec.sport.hockey       0.97      0.99      0.98       231
               sci.crypt       0.96      0.90      0.93       233
         sci.electronics       0.88      0.86      0.87       244
                 sci.med       0.95      0.89      0.92       256
               sci.space       0.92      0.92      0.92       246
  soc.religion.christian       0.93      0.98      0.95       252
      talk.politics.guns       0.74      0.89      0.81       249
   talk.politics.mideast       0.94      0.89      0.91       281
      talk.politics.misc       0.72      0.62      0.67       259
      talk.religion.misc       0.66      0.47      0.55       236

               micro avg       0.86      0.86      0.86      5000
               macro avg       0.86      0.86      0.86      5000
            weighted avg       0.86      0.86      0.86      5000


Score of inbuilt sklearn's MultinomialNB on the same data : 0.861
```

پایان