

پروژه نهایی ساختمان داده

نیمسال تحصیلی اول - سال 1402 - 1403

Decision Tree

مقدمه

یکی از پرکاربردترین الگوریتم‌های داده‌کاوی، الگوریتم درخت تصمیم است. در داده‌کاوی، درخت تصمیم یک مدل پیشبینی کننده است به طوری که می‌تواند برای هر دو مدل رگرسیون و طبقه‌ای مورد استفاده قرار گیرد. زمانی که درخت برای کارهای طبقه‌بندی استفاده می‌شود، به عنوان درخت طبقه‌بندی (Classification Tree) شناخته می‌شود و هنگامی که برای فعالیت‌های رگرسیونی به کار می‌رود درخت رگرسیون (Regression Decision Tree) نامیده می‌شود.

در این پروژه قصد داریم درخت تصمیم را برای طبقه بندی داده ها پیاده سازی کنیم.

توضیحات درخت طبقه بندی

در این درخت شما باید با استفاده داده‌های Train درخت خود را تشکیل بدهید و بعد از آن با داده‌های Test، خروجی داده‌ها را پیشبینی کنید. در اصل درخت تصمیم به عنوان یک تابع عمل می‌کند.

داده های Train شامل دو بخش ویژگی (attributes) و خروجی (labels) هستند.

داده های Test فقط شامل ویژگی ها هستند.

مفاهیم درخت تصمیم به صورت زیر است:

- هر گره داخلی نشان دهنده ویژگی های موجود هست و شاخه های خارج شده از گره نشان دهنده مقادیری هستند که آن ویژگی میتواند بگیرد.
- هر برگ نشان دهنده خروجی است.

آنتروپی (Entropy): آنتروپی معیاری برای محاسبه‌ی ناخالصی است که بین 0 و 1 متغیر است. هر چه آنتروپی یک ویژگی بالاتر باشد به این معنی است که آن خروجی آن متغیر مقدارهای مختلف تری می‌گیرد.

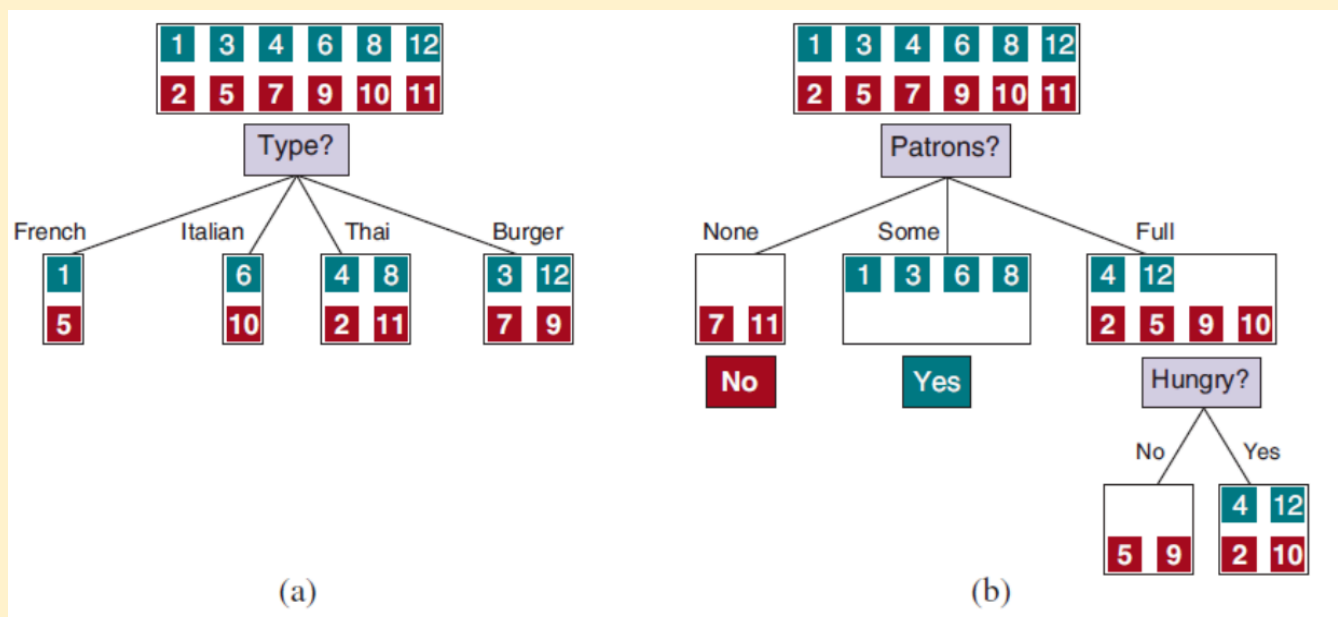
$$E(X) = - \sum_{x \in X} p(x) \log p(x)$$

که $P(x)$ نشان دهنده احتمال رخ دادن هر خروجی است.

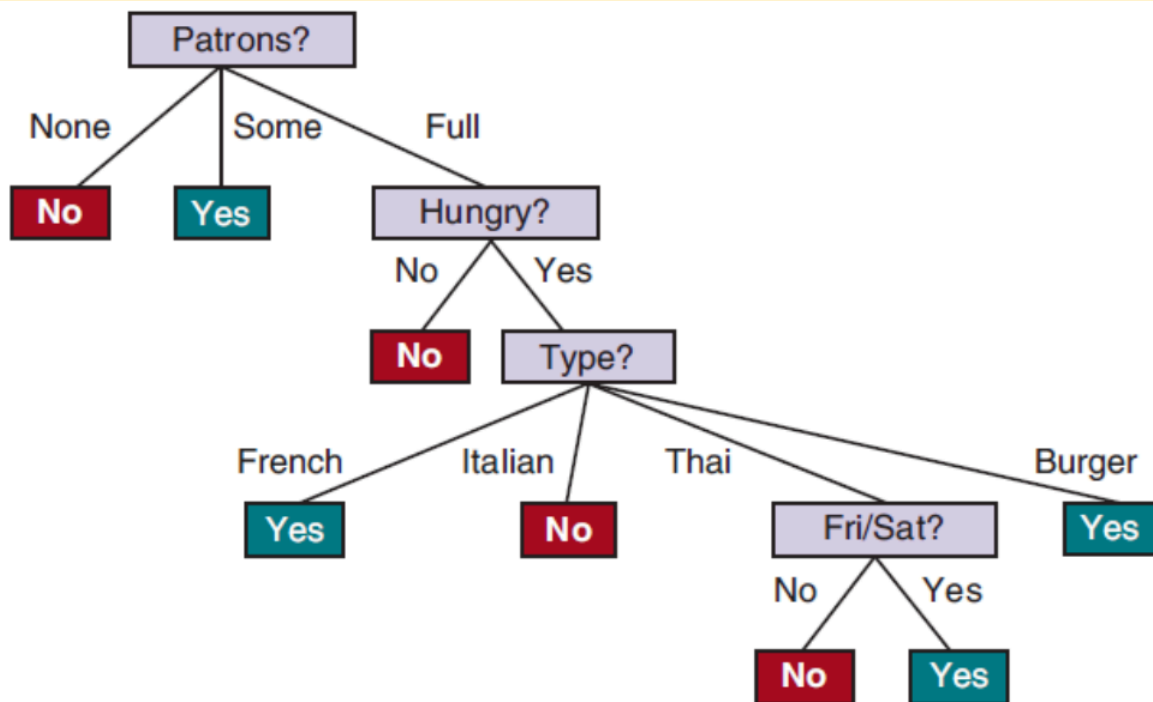
Information gain: برای اینکه در هر قسمت برای تقسیم داده ویژگی درست را انتخاب کنیم باید شاخصی داشته باشیم تا بر اساس آن بهینه‌ترین ویژگی را انتخاب کنیم.

$$IG = E(\text{parent}) - \sum w_i E(\text{child}_i)$$

از بین Gain های بدست آمده ویژگی با بالاترین Gain را برای تقسیم داده انتخاب می‌کنیم. اگر بخواهیم خلاصه تعریف کنیم، این مقیاس برای ما مشخص میکند که به واسطه‌ی انتخاب این ویژگی، چه قدر ویژگی‌هایمان به حد خوبی در خالص سازی داده‌هایمان به ما کمک خواهند کرد و چه قدر اطلاعات از انتخاب این ویژگی به دست می‌آوریم.



Example	Input Attributes										Output
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
x₁	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>0–10</i>	<i>y₁ = Yes</i>
x₂	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>30–60</i>	<i>y₂ = No</i>
x₃	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Some</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>y₃ = Yes</i>
x₄	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Thai</i>	<i>10–30</i>	<i>y₄ = Yes</i>
x₅	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>>60</i>	<i>y₅ = No</i>
x₆	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Italian</i>	<i>0–10</i>	<i>y₆ = Yes</i>
x₇	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>y₇ = No</i>
x₈	<i>No</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Thai</i>	<i>0–10</i>	<i>y₈ = Yes</i>
x₉	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>>60</i>	<i>y₉ = No</i>
x₁₀	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>Italian</i>	<i>10–30</i>	<i>y₁₀ = No</i>
x₁₁	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>0–10</i>	<i>y₁₁ = No</i>
x₁₂	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>30–60</i>	<i>y₁₂ = Yes</i>



1- کلاس Tree

```
class Tree{
    int depth;
    int getDepth();
    createTree(float[][] data, float[] labels);
    float Entropy(float[] labels);
    float iGain(int pEntropy, float[] weight, float[] entropies);
}
```

- متد `createTree`: آرایه دو بعدی از ویژگی ها دریافت میکند که هر آرایه داخل آن نشان دهنده مقدار های هر ویژگی آن متغیر است. همچنین آرایه تک بعدی دریافت می کند که خروجی متناظر با هر متغیر را نشان می دهد. در نهایت درخت تصمیم ساخته شده را برمی گرداند.
- متد `getDepth`: مقدار `depth` را بر می گرداند که نشان دهنده حداکثر عمق درخت است.
- متد `Entropy`: آرایه ای از خروجی ها دریافت و آنتروپی آن ها را محاسبه می کند.
- متد `iGain`: آنتروپی پدر، وزن فرزندان و آنتروپی متناظر با آن ها را دریافت می کند و `Gain` را محاسبه می کند

کلاس DecisionTreeClassifier

این کلاس درون خود از کلاس Tree استفاده می‌کند.

```
class DTreeClassifier {
    DTreeClassifier(float[][] data, float[] labels);
    float Predict(float[], int depth);
    float[] PredictAll(float[][] data, int depth);
    float accuracy(int[] labels, int[] labels_predicted);
}
```

- در هنگام ایجاد یک شیء از این کلاس، آرایه از ویژگی‌ها به همراه خروجی آن‌ها به متد سازنده کلاس پاس داده می‌شود.
- متد Predict: ویژگی‌ها را دریافت می‌کند و خروجی متناسب با آن را با استفاده از درخت مشخص می‌کند. depth مشخص می‌کند تا چه عمق درخت باید پیشبینی انجام شود.
- متد PredictAll: مثل متد Predict عمل می‌کند با این تفاوت که ویژگی‌های چند متغیر را دریافت و لیستی از خروجی‌های پیشبینی شده برمی‌گرداند.
- متد accuracy: لیستی از خروجی‌های اصلی و خروجی‌های پیشبینی شده دریافت و میزان دقت الگوریتم را برمی‌گرداند.

الگوریتم Random Forest

الگوریتم جنگل تصادفی Random Forest یک طبقه بندی است که شامل تعدادی درخت تصمیم در زیرمجموعه های مختلف مجموعه داده قرار دارد و برای بهبود دقت پیشبینی آن مجموعه داده، میانگین میگیرد. جنگل تصادفی به جای تکیه بر یک درخت تصمیم، پیشبینی را از هر درخت و براساس اکثریت آرا پیش بینی میکند و نتیجه نهایی را به عنوان خروجی در نظر میگیرد. تعداد بیشتر درختان در جنگل منجر به دقت بالاتری میشود و از بروز مشکل Overfitting جلوگیری میکند.

کلاس RandomForestClassifier

-دیتاهای قرار داده شده ویژگی های بیمارانی است که باید نوع دیابت آن ها را مشخص کنید. با پیاده سازی درست دقت حدود 70% تا 80% را می توانید به دست بیارید.

موارد نمره اضافه:

پیاده سازی رندوم فارست.
درخت حاصل را به صورت گرافیکی نمایش دهید.

نکات تکمیلی:

- استفاده از هر زبان برنامه نویسی برای انجام پروژه مجاز است.
 - در صورت نیاز می توانید متدهای کمکی پیاده سازی کنید.
 - استفاده از هرگونه کتابخانه خارج از کتابخانه های اصلی زبان، مجاز نیست.
 - در صورت نیاز به استفاده از دیگر ساختمانهای داده، پیاده سازی آنها الزامی است.
 - پروژه به صورت *انفرادی* یا در *گروه های 2 نفره* قابل انجام است (آپلود توسط هر دو عضو الزامی است).
 - گروه ها می بایست از گیت استفاده کنند.**
 - هنگام تحویل، هر دو عضو گروه باید تسلط کامل داشته باشند.
 - در صورت مشاهده شباهت غیر متعارف میان پروژه افراد، نمره -100 برای هر دو نفر در نظر گرفته میشود.
 - تسلط به بخشهای مختلف پروژه در هنگام تحویل الزامی است.
 - فایل های نهایی پروژه خود را در قالب زیر در سامانه VU بارگزاری کنید:
- FirstNameLastNames_StudentNumbers_PR2.zip**
- برای توضیحات بهتر می توانید لینک [1](#) و [2](#) را مشاهده کنید.

موفق باشید