

# 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. هر چه آنتروپی یک مجموعه بالاتر باشد به این معنی است که آن مجموعه مقادیر مختلف تری نسبت به مجموعه ای که آنتروپی پایین‌تری دارد، در خودش دارد.

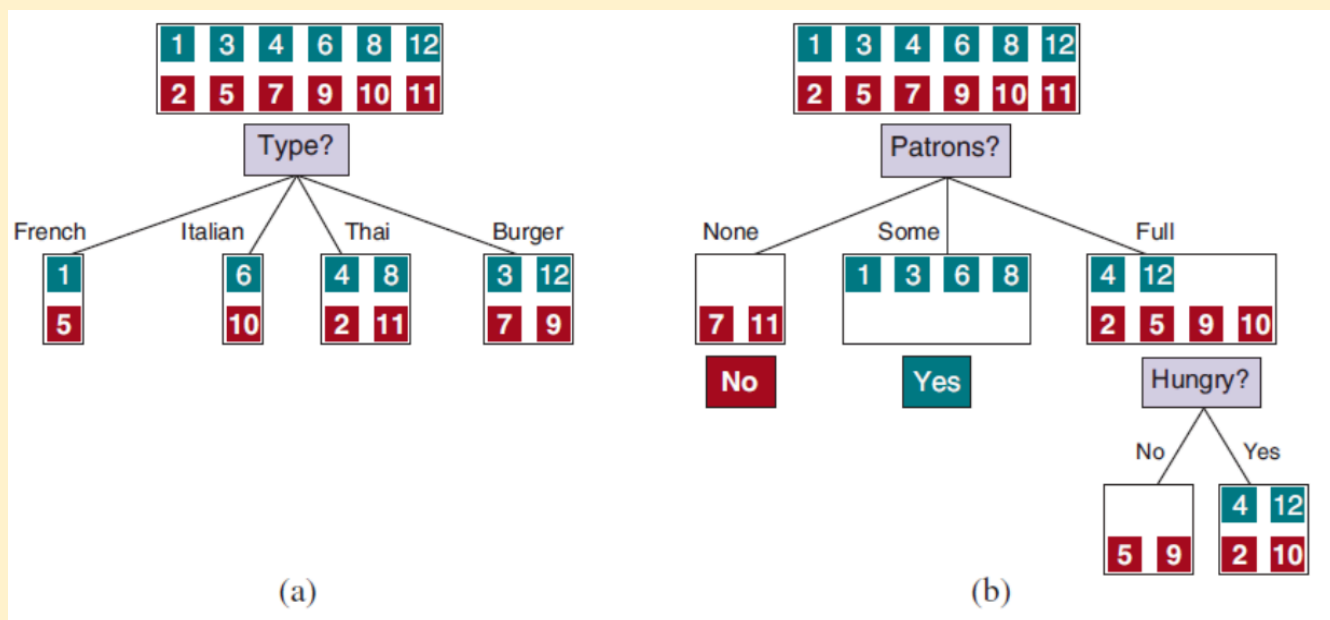
$$E(X) = - \sum_{x \in X} p(x) \log p(x)$$

که  $P(x)$  نشان دهنده احتمال رخ دادن هر عنصر داخل مجموعه است.

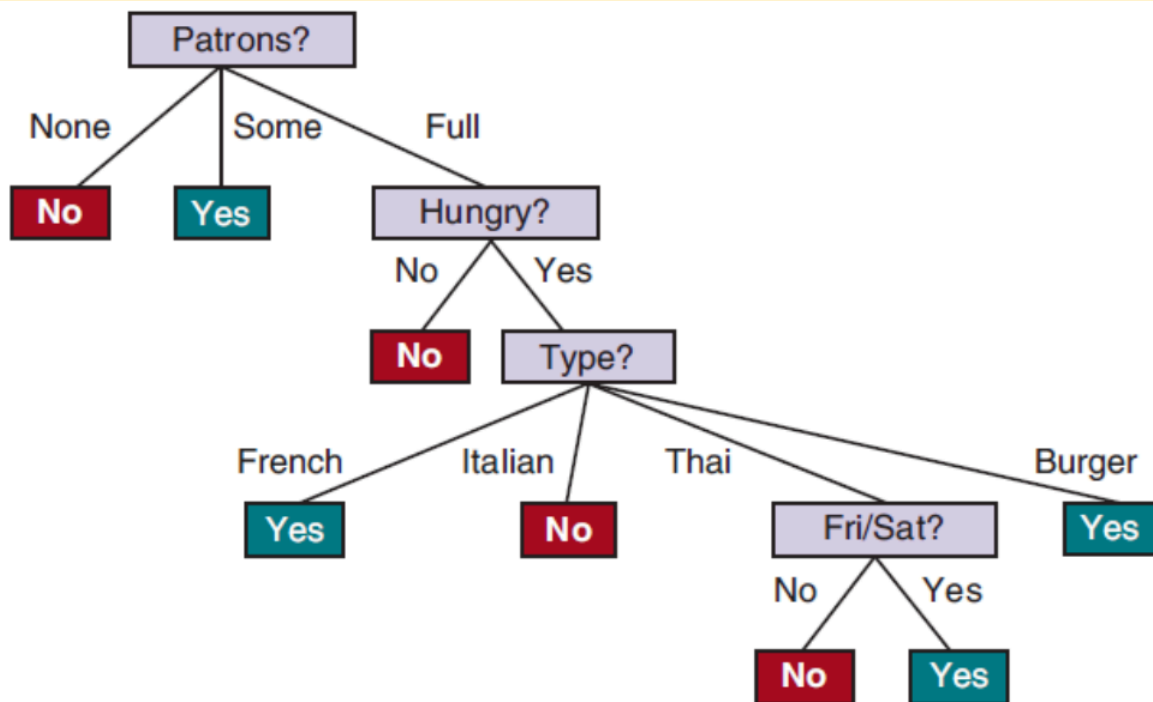
**Information gain:** برای اینکه در هر قسمت برای تقسیم داده ویژگی درست را انتخاب کنیم باید شاخصی داشته باشیم تا بر اساس آن بهینه‌ترین ویژگی را انتخاب کنیم.

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

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>
<b>x<sub>1</sub></b>	<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>y<sub>1</sub> = Yes</i>
<b>x<sub>2</sub></b>	<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>y<sub>2</sub> = No</i>
<b>x<sub>3</sub></b>	<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>y<sub>3</sub> = Yes</i>
<b>x<sub>4</sub></b>	<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>y<sub>4</sub> = Yes</i>
<b>x<sub>5</sub></b>	<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>&gt;60</i>	<i>y<sub>5</sub> = No</i>
<b>x<sub>6</sub></b>	<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>y<sub>6</sub> = Yes</i>
<b>x<sub>7</sub></b>	<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>y<sub>7</sub> = No</i>
<b>x<sub>8</sub></b>	<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>y<sub>8</sub> = Yes</i>
<b>x<sub>9</sub></b>	<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>&gt;60</i>	<i>y<sub>9</sub> = No</i>
<b>x<sub>10</sub></b>	<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>y<sub>10</sub> = No</i>
<b>x<sub>11</sub></b>	<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>y<sub>11</sub> = No</i>
<b>x<sub>12</sub></b>	<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>y<sub>12</sub> = 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 یا Random Forest یک متد برای طبقه بندی داده ها است که بر اساس مجموعه ای از درختان تصمیم که به صورت تصادفی ساخته شده اند، عمل می کند. هر درخت تصمیم در این الگوریتم بر روی یک زیر مجموعه تصادفی از داده ها آموزش داده شده و سپس برای پیش بینی یک نمونه جدید، پیش بینی همه درختان گرفته می شود و برجستگی که بیشترین رای را داشته باشد به عنوان پیش بینی نهایی ارائه می شود. Random Forest به دلیل استفاده از تعداد زیادی درخت تصمیم، معمولاً عملکرد بهتری نسبت به یک درخت تصمیم تکی دارد و همچنین قابلیت کنترل بیش برآزش (overfitting) را نیز دارد. برای توضیحات و مثال بیشتر به لینک زیر مراجعه کنید:

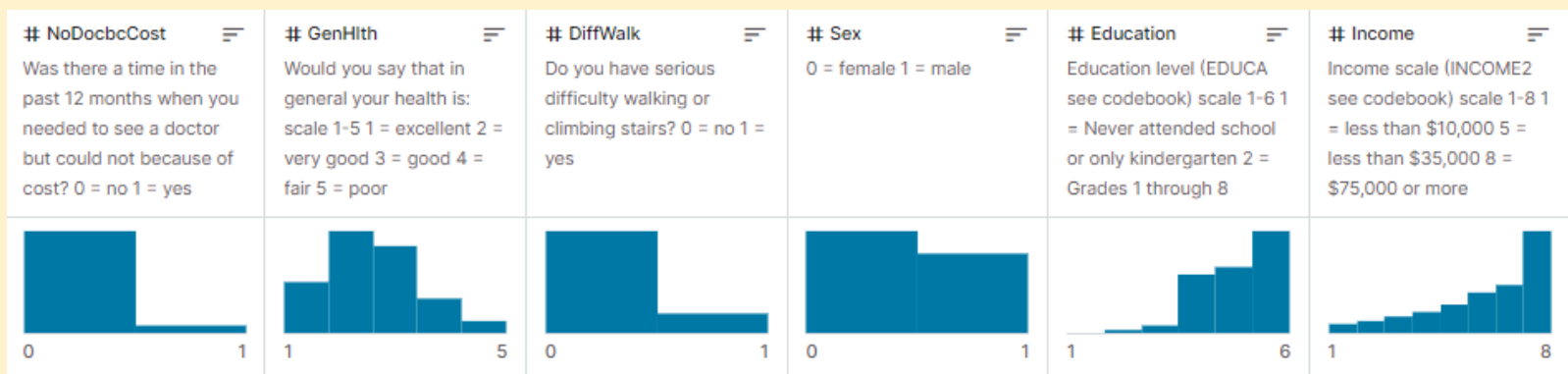
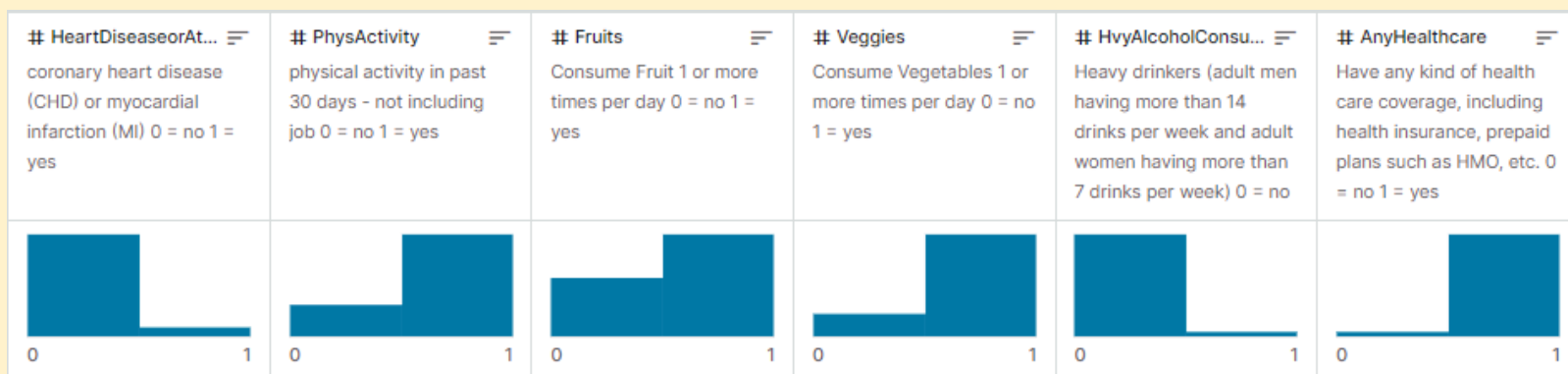
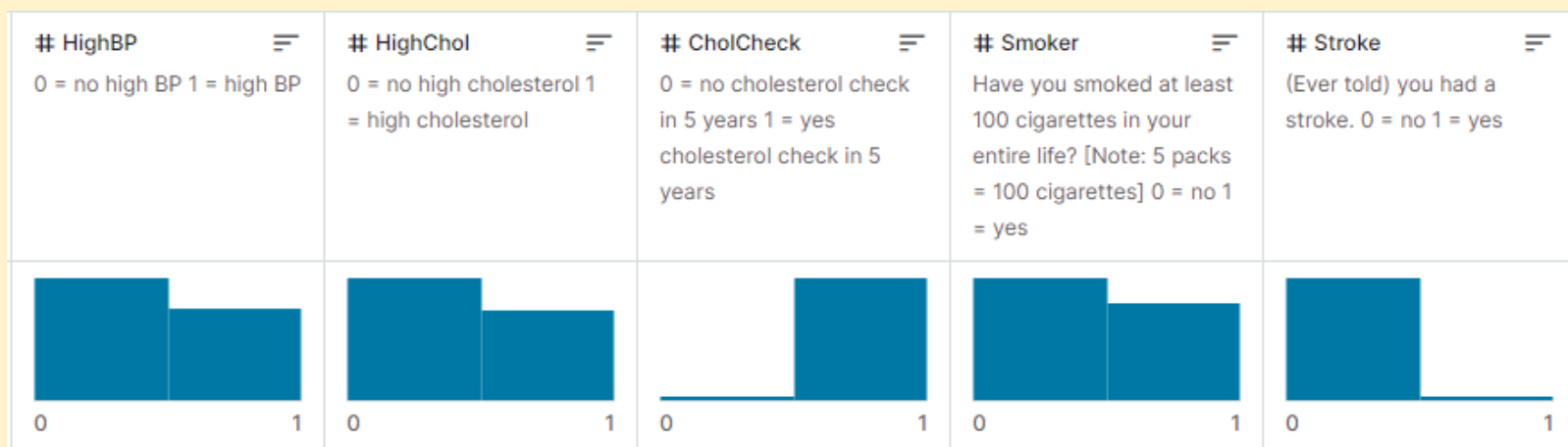
[Understand Random Forest Algorithms With Examples](#)

## کلاس RandomForestClassifier

```
class RandomForestClassifier{
    RandomForestClassifier(float[][] data, float[] labels, int number_of_estimators, int max_depth);
    float Predict(float[] data, int depth);
    float PredictAll(float[][] data, int depth);
    float Accuracy(float[] labels , float[] predicted_labels);
}
```

- تنها تفاوت این کلاس با کلاس DecisionTreeClassifier در number\_of\_estimator است که همان تعداد درختان جنگل برای تصمیم گیری درباره پیشبینی داده وارد شده خواهد بود.
- دقت کنید که در کلاس RandomForestClassifier هر درخت تصمیم باید از یک زیر مجموعه تصادفی از داده ها تشکیل شود و سپس پیشبینی ای که بیشترین رای را در بین همه درختان تصمیم داشت به عنوان پیشبینی نهایی انتخاب شود.

-دیتاهای قرار داده شده ویژگی های بیمارانی است که باید نوع دیابت آن ها را مشخص کنید. با پیاده سازی درست دقت حدود 75% تا 85% را می توانید به دست بیاورید.



1. فایل `feature_train` شامل  $n$  سطر که هر سطر نشان دهنده یک متغیر است و شامل مقادیری است که به ازای هر ویژگی موجود اختیار کرده است. (نام ویژگی ها در سطر اول موحود است)
2. فایل `label_train` این فایل شامل  $n$  سطر است که هر سطر نوع دیابت را نشان می‌دهد. (به عنوان مثال شخصی در سطر پنجم فایل `feature_train` که ویژگی هایش مشخص شده، نوع دیابتش در سطر پنجم `label_train` به شما داده شده.)
3. فایل `feature_test` این فایل شامل  $m$  سطر است که هر سطر نشان دهنده یک متغیر است و شامل مقادیری است که به ازای هر ویژگی موجود اختیار کرده است. از این فایل برای تست دقت الگوریتم استفاده کنید.
4. فایل `label_test` این فایل شامل  $n$  سطر است که هر سطر نوع دیابت را نشان می‌دهد. از این فایل برای تست دقت الگوریتم استفاده کنید.

## 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 می‌توانید متدهای کمکی پیاده‌سازی کنید.
- استفاده از هرگونه کتابخانه خارج از کتابخانه‌های اصلی زبان، مجاز نیست.
- در صورت نیاز به استفاده از دیگر ساختمانهای داده، پیاده‌سازی آنها الزامی است.
- پروژه به صورت *انفرادی* یا در *گروه‌های 2 نفره* قابل انجام است (آپلود توسط هر دو عضو الزامی‌ست).
- گروه‌ها می‌بایست 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.
- فایل‌های نهایی پروژه خود را در قالب زیر در سامانه VU بارگذاری کنید:



FirstNameLastNames\_StudentNumbers\_PR2.zip

برای توضیحات بهتر می‌تونید لینک [1](#) و [2](#) را مشاهده کنید.

**موفق باشید**