سوال ۱ –

قسمت الف) اصل اول occam's razor نام دارد. این اصل میگه پس از ایجاد یک فرضیه، تا جایی که امکانش هست جزئیات و پیچیدگی هارا حذف میکنیم تا فرضیه ساده تر بشه به طوری که سازگاری فرضیه و مدل با داده ها از دست نره.یعنی ساده ترین مدل که با داده ها fit بشه، بهترین مدل است. پیچیدگی هم دو نوع است که البته با هم در ارتباط مستقیم اند مگر در حالت استثنا(SVM). نوع اول پیچیدگی h است و نوع دوم پیچیدگی H که در این اصل، پیچیدگی h بیان میشود اما اثبات این اصل روی پیچیدگی H صحبت میکند.

اصل دوم sampling bias نام دارد. این نوع بایاس در اصل بایاس نمونه برداری است.در این حالت داده های آموزشی با داده های آزمون توزیع یکسانی ندارند و مدل خوب جواب نمیدهد. راه حل آن هم وزن دار کردن خطا بر اساس اختلاف توزیع داده های آموزشی و آزمون است.

اصل سوم data snooping است که میگه داده ها نباید روی پروسه یادگیری تاثیری بگذارند که بعدا بتوانند قضاوت خوبی روی مدل ایجاد شده انجام دهند. در اصل به بیان ساده یعنی یادگیری نباید حتی بخشی از آن در ذهن اتفاق بیفتد(برای مثال انتخاب شکل و درجه مدل با vision سازی داده ها)

قسمت ب) اغلب این قانون برقرار است اما نه همیشه. همان طور که گفته شد، در SVM ، مرز تصمیم گیری میتواند درجه بالا و پیچیده باشد(پیچیدگی h) اما بخاطر کم بودن تعداد SV ها ، پیچیدگی H کم باشد و VC-dim نیز در نتیجه کم باشد لذا سادگی در اصل occam's vector حفظ شود. برای مثال یک خط جدا کننده SVM را در نظر بگیرید که درجه ۳ یا حتی ۴ باشد اما در کل ما ۳ عدد SVM داشته باشیم.

قسمت ج) این دو اصل در حقیقت مفاهیم جدایی دارند زیرا data snooping صحبت اش راجع به اثر دیتاست روی یادگیری و در نهایت آزمایش مدل ایجاد شده با آن دیتاست است. اما sampling bias صحبت اش تاثیر چگونگی نمونه برداری و تاثیر داده های انتخاب شده روی نتیجه آزمون مدل است. لذا این دو ارتباطی باهم ندارد. اما در موارد خاص، به خصوص در تجارت و سهام و اینگونه مسائل، این دو میتوانند کمی به هم مربوط شوند. برای مثال فرض کنیم سیستمی ایجاد کرده ایم که با تحلیل قیمت های ارز های دیجیتال نوپا در گذشته، میخواهد تشخیص دهد چه ارزی را خریداری کنیم و پول را روی آن ریسک کنیم. توجه شود ارز های دیجیتال نوپا(استارت آپ) امکان ورشکستگی دارند. یعنی ممکن است پولی را سرمایه گذاری کنیم و پس از مدتی شرکت مربوطه ورشکسته شود و استارت آپ با شکست مواجه شود و کل پول شما از بین برود. حال مشکلی که هست این است که ما داده های ارزهایی را داریم که در حال حاضر ورشکسته نشده اند و این یعنی

مشکل sampling bias زیرا داده های ارز هایی که در گذشته ورشکست کرده اند را نداریم و در موقع تست کردن مدل ایجاد شده، چون توزیع داده های آموزشی با آزمایشی متفاوت است، غافلگیر شده و مدل ما شکست میخورد در صورتی که موقع validation مدل ما خوب کار میکرد.حال اگر دنبال شرکت هایی بگردیم که در گذشته ورشکست کرده اند و با اینکار بخواهیم داده های آموزشی خودمان را جوری تغییر دهیم که توزیع آموزش و آزمایش برابر شود، داریم data snooping میکنیم زیرا ما حق نداشتیم با توجه به نتایج و خروجی ها، داده ها را بر طبق میل خودمان تغییر دهیم و قسمتی از یادگیری را توی ذهن خود انجام دهیم و داده هارا مستقل از خروجی بکنیم.

سوال ۲ –

قسمت الف) در locally weighted linear regression مسئله بهینه سازی ای که داریم، مسئله زیر است:

$$minimize_{\theta} \quad \sum_{i=1}^{m} w^i (y^i - \theta^T x^i)^2$$

حال برای سادگی اثبات، فرض میکنیم در فضای ۱ بعدی هستیم و $\theta = [\theta_0, \theta_1]$ و x و y بردار هایی با طول m هستند.

حال تابع هزینه برابر است با:

$$J(\theta) = \sum_{i=1}^{m} w^i \left(y^i - (\theta_0 + \theta_1 x^i)\right)^2$$

حال مشتق هارا محاسبه میکنیم:

$$\frac{\partial J}{\partial \theta_0} = -2 \sum_{i=1}^{m} w^i \left(y^i - (\theta_0 + \theta_1 x^i)\right)$$

$$\frac{\partial J}{\partial \theta_1} = -2 \sum_{i=1}^{m} w^i \left(y^i - (\theta_0 + \theta_1 x^i)\right) x^i$$

اگر مشتق هارا برای بهینه سازی برابر با صفر قرار دهیم، داریم:

$$\frac{\partial J}{\partial \theta_0} = -2 \sum_{i=1}^{m} w^i \left(y^i - (\theta_0 + \theta_1 x^i)\right) = 0$$

$$\rightarrow \sum_{i=1}^{m} w^i \theta_0 + \sum_{i=1}^{m} w^i \theta_1 x^i = \sum_{i=1}^{m} w^i y^i \qquad \text{معادله} (1)$$

$$\frac{\partial J}{\partial \theta_1} = -2\sum_{i=1}^{m} w^i \left(y^i - (\theta_0 + \theta_1 x^i)\right) x^i = 0$$

$$\rightarrow \sum_{i=1}^{m} w^i \theta_0 + \sum_{i=1}^{m} w^i \theta_1 x^i x^i = \sum_{i=1}^{m} w^i y^i x^i \qquad \text{معادله}(2)$$

اگر رابطه ۱ و ۲ را باهم به شکل ماتریسی بنویسیم، داریم:

$$\begin{bmatrix} \sum w^i & \sum w^i x^i \\ \sum w^i x^i & \sum w^i x^i x^i \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} \sum w^i y^i \\ \sum w^i y^i x^i \end{bmatrix}$$

ماتریس ۲*۲ سمت چپ، در اصل همان $X^T WX$ است(آن را A مینامیم) و ماتریس ۱*۲ سمت راست مساوی هم $X^T WY$ است(آن را B مینامیم). لذا با ضرب دو طرف در معکوس ماتریس ۲*۲ داریم:

$$A\theta = B \rightarrow \ \theta = A^{-1}B = \ (X^T WX)^{-1} X^T WY$$

قسمت ب)در *linear regression* معمولی، به هر نقطه اطراف نقطه مورد پیش بینی ما، وزن یکسانی میدهد و به یک اندازه به آن ها توجه میشود و در محاسبه تابع هزینه تاثیر میدهد و مدل را با آن نقاط ***fit*** میکند لذا توابع پیچیده و غیرخطی را خوب نمیتواند مدل و پیش بینی کند. اما در ***locally weighted regression*** ، با نگاه محلی به نقاط اطراف نقطه مورد نظر ما، میتوانیم مدل های خطی ای را ***fit*** کنیم در محل نقطه مورد نظرمان و با اینکار پیش بینی روی فضای غیرخطی و پیچیده را راحت تر کنیم.بخاطر اینکه وزن هر نقطه متفاوت است(مثلا در این قسمت گفته شده است وزن گاوسی استفاده میشود) لذا وقتی نقطه همسایه به نقطه مورد پیش بینی ما نزدیک باشد، وزن آن با توجه به رابطه زیر نزدیک به ۱ میشود(چون مرکز توزیع همان نقطه مورد پیش بینی است، لذا ارتفاع توزیع نسبت مستقیمی با بزرگی وزن آن نقطه دارد) و برعکس اگر از نقطه دور باشند، وزن آن نزدیک به صفر میشود. لذا در این نوع رگراسیون، به نقاط همسایه و نزدیک نقطه مورد نظر که میخاهیم برچسب آن را پیش بینی کنیم، توجه بیشتری میشود و در محاسبات، هزینه مورد نظر آن بیشتر تاثیر میگذارد لذا مدل با آن نقاط بیشتر ***fit*** میشود. هرچه از نقطه مورد پیش بینی هم دور شویم، این تاثیر و ***fit*** شدن مدل کمتر میشود.

اگر وزن ها توزیع گاوسی داشته باشند، رابطه آن به شکل زیر درمی‌آید:

$$w^i = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\|x^i - x\|^2}{2\sigma^2}\right)$$

لذا واریانس وزن ها نقش تنظیم *smoothness* مدل را دارند(*bandwidth parameter*).هرچه واریانس بیشتر باشد، مدل ایجاد شده *smooth* تر میشود زیرا توزیع گاوسی پهن تر میشود.

خیر لازم نیست واریانس توزیع برای تمام داده ها یکسان باشد زیرا با متفاوت در نظر گرفتن واریانس در محل های متفاوت داده ها میتوانیم توزیع گاوسی مختلف داشته باشیم(چیزی شبیه *RBF*) و هر نقطه تحت تاثیر یک توزیع با واریانس متفاوت میتوانند قرار بگیرد و بنا به *dense* بودن یا نبودن نقاط اطراف آن، با اینکار میتوانیم *fit* بهتری داشته باشیم.

سوال ۳-

قسمت الف) ابتدا این الگوریتم را برای مسئله رگراسیون توضیح میدهیم: فرض کنیم یک مجموعه داده به شکل زیر داریم:

$$(x_1, y_1), \dots, (x_n, y_n)$$

مسئله رگراسیون یعنی یک مدل *F(x)* را با داده ها *fit* کنیم که تابع هزینه(در رگراسیون از *square loss* استفاده میشود) را مینیموم کنیم.روند *gradient boosting* اینجوریه که با یک مدل اولیه ساده شروع کرده و چون به اندازه کافی خوب پیش بینی نمیکند، هر دفعه به مدل قبلی(اغلب درخت) یک مدل جدید(درخت رگراسیون دیگر) اضافه میکنیم تا دقت پیش بینی بیشتر شود(*loss* کمتر شود). یعنی در مرحله دوم ، مسئله به شکل زیر است:

$$F(x_1) + h(x_1) = y_1$$

$$\dots$$

$$F(x_n) + h(x_n) = y_n$$

که *F* همان مدل اولیه و *h* همان مدل ثانویه که به مدل اولیه اضافه میشود است.

در حقیقت مدل *h* ای را باید بدست آوریم که:

$$h(x_1) = y_1 - F(x_1)$$

$$\dots$$

$$h(x_n) = y_n - F(x_n)$$

پس باید درخت رگراسیون *h* را با داده های زیر *fit* کنیم:

$$\big(x_1, y_1 - F(x_1)\big), \dots, \big(x_n, y_n - F(x_n)\big)$$

به $y_i - F(x_i)$ ها *residual* میگویند که یعنی جاهایی که مدل *F* خوب کار نمیکنند.حال اگر مدل جدید *F+h* به اندازه کافی دقت لازم را نداشت، این روند را ادامه میدهیم یعنی یک درخت رگراسیون دیگر مثل *h2* را به مدل آخری اضافه

میکنیم و همین روند را آنقدر تکرار میکنیم، که به دقت کافی برسیم.حال برای سادگی روابط، نشان میدهیم که ***residual*** ها دقیقا همان مشتق تابع هزینه در جهت خلاف گرادیان است:

$$L\big(y, F(x)\big) = \frac{\big(y - F(x)\big)^2}{2}$$

ما میخواهیم J را با تنظیم مناسب $F(x_1), F(x_2), \dots, F(x_n)$ کمینه کنیم:

$$J = \sum_i L(y_i, F(x_i))$$

حال برای محاسبه مشتق ها داریم:

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = F(x_i) - y_i \to y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

پس:

$$F_{j+1}(x_i) = F_j(x_i) - 1\frac{\partial J}{\partial F(x_i)} \to \ regression\ with\ square\ loss \cong gradient\ descent$$

اگر به جای ***residual*** ها از ***negative gradient*** استفاده کنیم، الگوریتم ***gradient boosting for regression*** :

$$-g(x_i) = -\frac{\partial L\big(y_i, F(x_i)\big)}{\partial F(x_i)} = y_i - F(x_i)$$

ابتدا با یک مدل اولیه به شکل زیر شروع میکنیم:

$$F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$$

مراحل زیر را آنقدر تکرار میکنیم که مدل همگرا شود:

۱- محاسبه گرادیان منفی $-g(x_i)$

۲- درخت رگراسیون h را با گرادیان منفی بالا ***fit*** میکنیم.

۳- $F_{j+1} = F_j + h$

حال مسئله ***gradient boosting*** را برای دسته بندی تعریف میکنیم:

اگر فرض کنیم میخواهیم برچسب یک سری تصاویر از دستنویس های انسانی را پیش بینی کنیم(برای مثال حروف الفبای انگلیسی) پس ۲۶ کلاس داریم و مدل ما ۲۶ تا تابع ***score*** به شکل زیر میشود:

$$F_A, F_B, \dots, F_Z$$

اگر برای محاسبه احتمال تعلق به کلاس ها، از ***score*** ها به شکل ***softmax*** استفاده کنیم، لذا:

$$P_A(x) = \frac{e^{F_A(x)}}{\sum_{c=A}^{Z} e^{F_c(x)}}$$

$$\dots$$

$$P_Z(x) = \frac{e^{F_Z(x)}}{\sum_{c=A}^{Z} e^{F_c(x)}}$$

بنابراین، کلاس پیش بینی شده برابر است با کلاسی که بیشتر احتمال را دارد. لذا الگوریتم به شکل زیر میشود:

۱- تبدیل برچسب داده i ام به یک توزیع احتمالاتی واقعی $Y_c(x_i)$ برای مثال اگر برچسب داده ۳ ام H باشد:

$$Y_A(x_3) = 0, \dots, Y_H(x_3) = 1, \dots, Y_Z(x_3) = 0$$

۲- بر اساس مدل های $F_A, F_B, \dots, F_Z$ ، توزیع احتمال پیش بینی شده $P_c(x_i)$ را محاسبه میکنیم.

۳- محاسبه اختلاف بین توزیع احتمال واقعی و پیش بینی شده به عنوان خطا

۴- هر بار با تغییر مدل های F سعی میکنیم تابع خطا را کمینه کنیم.

لذا تفاوت این متد در رگراسیون و دسته بندی، به غیر از تابع هزینه آن و محاسبات مربوطه، این است که در رگراسیون با اضافه کردن درخت رگراسیون به مدل سعی در بهبود مدل داریم اما در دسته بندی، با تغییر توزیع احتمالاتی سعی در کاهش خطا و بهتر پیش بینی کردن مدل در راند های مختلف داریم.

قسمت ب) داریم:

$$F_m(x) = F_{m-1}(x) + h_m(x) = y$$

$$\rightarrow h_m(x) = y - F_{m-1}(x)$$

$$L_{cross-entropy} = -\sum_i y_i \log\left(F(x_i)\right)$$

$$\rightarrow \frac{\partial L_{cross-entropy}}{\partial F} = F(x) - y = -h_m(x) = -residual \quad ***$$

$$\hat{F} = argmin_F \, E_{x,y}[L(y, F(x)]$$

$$\hat{F}(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + C$$

$$F_m(x) = F_{m-1}(x) + argmin_{h_m}\left[\sum_{i=1}^{n} L\big(y_i, F_{m-1}(x_i) + h_m(x_i)\big)\right]$$

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum\nolimits_{i=1}^{n} \nabla_{F_{m-1}} L\big(y_i, F_{m-1}(x_i)\big)$$

$$\rightarrow \gamma_m = argmin_\gamma \sum\nolimits_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) - \gamma \nabla_{F_{m-1}} L\big(y_i, F_{m-1}(x_i)\big)\right)$$

$$***\rightarrow \gamma_m = argmin_\gamma \sum\nolimits_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

برای حل این مسئله، ابتدا γ_1 را محاسبه میکنیم:

$$\gamma_1 = argmin_\gamma \; -y_1 \log\big(F_{m-1}(x_i) + \gamma h_m(x_i)\big)$$

$$\rightarrow \frac{\partial}{\partial \gamma}\left[-y_1 \log\big(F_{m-1}(x_i) + \gamma h_m(x_i)\big)\right] = \frac{\partial}{\partial \gamma}[L(y_1, F_{m-1}(x_1) + \gamma h_m(x_i)] = 0$$

از آنجایی که مشتق بالا سخت است، از بسط تیلور برای تخمین تابع هزینه استفاده میکنیم:

$$L(y_1, F_{m-1}(x_1) + \gamma h_m(x_i)$$

$$\approx L\big(y_1, F_{m-1}(x_1)\big) + \frac{\partial}{\partial F}\big(y_1, F_{m-1}(x_1)\big)\big(\gamma h_m(x_1)\big) + \frac{1}{2}\frac{\partial^2}{\partial F^2}\big(y_1, F_{m-1}(x_1)\big)\big(\gamma h_m(x_1)\big)^2$$

پس برای محاسبه مقدار بهینه گاما، از تقریب بالا استفاده کرده:

$$\frac{\partial}{\partial\gamma}\left(\sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right)\right) = 0$$

$$\rightarrow \frac{\partial}{\partial\gamma}\left(L\left(y_1, F_{m-1}(x_1)\right) + \frac{\partial}{\partial F}\left(y_1, F_{m-1}(x_1)\right)\left(\gamma h_m(x_i)\right) + \frac{1}{2}\frac{\partial^2}{\partial F^2}\left(y_1, F_{m-1}(x_1)\right)\left(\gamma h_m(x_i)\right)^2\right)$$

$$= \sum_{i=1}^{n}\left(\frac{\partial}{\partial F}\left(y_i, F_{m-1}(x_i)\right)\left(h_m(x_i)\right) + \frac{\partial^2}{\partial F^2}\left(y_i, F_{m-1}(x_i)\right)h_m(x_i)\gamma_m\right)$$

$$\rightarrow \gamma_m = -\frac{\sum_{i=1}^{n}\frac{\partial}{\partial F}\left(y_i, F_{m-1}(x_i)\right)\left(h_m(x_i)\right)}{\sum_{i=1}^{n}\frac{\partial^2}{\partial F^2}\left(y_i, F_{m-1}(x_i)\right)\left(h_m(x_i)\right)^2}$$

مشخص است که صورت کسر همان مجموع ***residual*** هاست(*) لذا با در نظر گرفتن باینری کراس انتروپی داریم:

$$p = \frac{e^{\ln(odds)}}{1 + e^{\ln(odds)}}$$

$$L_{binary-cross-entropy}(y_i, p) = -(y_i \ln(p) + (1 - y_i)\ln(1 - p))$$

$$= -\left(y_i \ln\left(\frac{p}{1-p}\right) + \ln(1-p)\right)$$

$$= -\left(y_i \ln(odds) - \ln\left(1 + e^{\ln(odds)}\right)\right.$$

$$= \ln\left(1 + e^{\ln(odds)}\right) - y_i \ln(odds) = \ln\left(1 + e^{F_{m-1}(x_i)}\right) - y_i \ln(F_{m-1}(x_i))$$

از طرفی مخرج کسر مربوطه به طور مشابه و با توجه به روابطی که بدست آوردیم، برابر است با:

$$\frac{\partial^2}{\partial F^2}\left(y_i, F_{m-1}(x_i)\right)h_m(x_i) = p_i(1 - p_i)h_m(x_i)$$

لذا با جایگذاری صورت و مخرج داریم:

$$\gamma_m = \frac{\sum_{i=1}^{n} y_i - p_i}{\sum_{i=1}^{n} p_i(1 - p_i)h_m(x_i)}$$

قسمت پ) ابتدا باید داده هارا ***preprocess*** کنیم(نرمالایز ، عددی کردن ستون های غیر عددی و ***dummy*** کردن ستون رنگ مورد علاقه):

Likes popcorn	Age	FC - B	FC - G	FC - R	Loves Troll 2
1	0.138	1	0	0	1
1	1	0	1	0	1
0	0.505	1	0	0	0
1	0.218	0	0	1	0
0	0.367	0	1	0	1
0	0.160	1	0	0	1

$$L_{CE}(y_i, p) = \ln\left(1 + e^{\ln(odds)}\right) - y_i \ln\,(odds)$$

$$\sum_{i=1}^{n} L(y_i, p) = 4\ln(odds) - 6\ln\left(1 + e^{\ln(odds)}\right)$$

$$F_0(x) = \frac{\partial}{\partial \gamma}\sum_{i=1}^{n} L(y_i, p) = -\left(4 - \frac{6e^{\ln(odds)}}{1 + e^{\ln(odds)}}\right) = 0 \rightarrow p \approx 0.67$$

$$\ln(odds) = \ln\left(\frac{p}{1-p}\right) = \ln(2) = 0.69 = F_0(x)$$

تکرار اول)

$$residual_1 = 1 - \frac{2}{3} = 0.33$$

$$residual_2 = 1 - \frac{2}{3} = 0.33$$

$$residual_3 = 0 - \frac{2}{3} = -0.67$$

$$residual_4 = 0 - \frac{2}{3} = -0.67$$

$$residual_5 = 1 - \frac{2}{3} = 0.33$$

$$residual_6 = 1 - \frac{2}{3} = 0.33$$

$$W_1 = \begin{bmatrix} -0.0007 \\ 0.0011 \\ 0.2502 \\ 0.2497 \\ 0.5000 \end{bmatrix}$$

$$b_1 = 0.75$$

$$s_1 = W_1^T + b_1 = \begin{bmatrix} 0.9995 \\ 1 \\ 1 \\ 0.2494 \\ 1 \\ 1 \end{bmatrix} , p_1 = \frac{1}{1+e^{-s_1}} = \begin{bmatrix} 0.7309 \\ 0.7310 \\ 0.7311 \\ 0.5620 \\ 0.7310 \\ 0.7311 \end{bmatrix}$$

$$\ln(odds) = \ln\left(\frac{p_1}{1-p_1}\right) = s_1$$

از قسمت ب همین سوال، داشتیم:

$$\gamma_1 = \frac{\sum_{i=1}^{n} y_i - p_i}{\sum_{i=1}^{n} p_i(1-p_i)h_m(x_i)} = \frac{0.33 + 0.33 - 0.67 - 0.67 + 0.33}{\dots} = \frac{0}{\dots} = 0$$

اگر همین منوال را برای بقیه داده ها جلو برویم، همواره مقدار گاما صفر میشود چون همواره برچسب درست پیش بینی میشود.بنابرین مدل نیازی به گسترش ندارد و *base learner* خروجی میشود. شاید هم من اشتباه میکنم!!!!

قسمت ت) با در نظر گرفتن درخت اولیه با یک گره با احتمال *pr=0.5* بخاطری دسته بندی باینری مسئله، داریم:

منبع: سایت statsquest

Likes popcorn	Age	Favorite color	Loves Troll 2	P1	R1
Yes	12	Blue	Yes	0.5	0.5
Yes	87	Green	Yes	0.5	0.5
No	44	Blue	No	0.5	-0.5
Yes	19	Red	no	0.5	-0.5
No	32	Green	Yes	0.5	0.5
No	14	Blue	Yes	0.5	0.5

روابط مربوطه را مینویسم اما محاسبات را دیگر تایپ نکرده و دستی حساب کردم و نتیجه نهایی را مینویسم:

$$Score = \frac{[\sum_{i=1}^{n} R_i]^2}{\sum_{i=1}^{n} p_i(1-p_i)} \quad , \quad W_{leaf} = \frac{\sum_{i=1}^{n} R_i}{\sum_{i=1}^{n} p_i(1-p_i)}$$

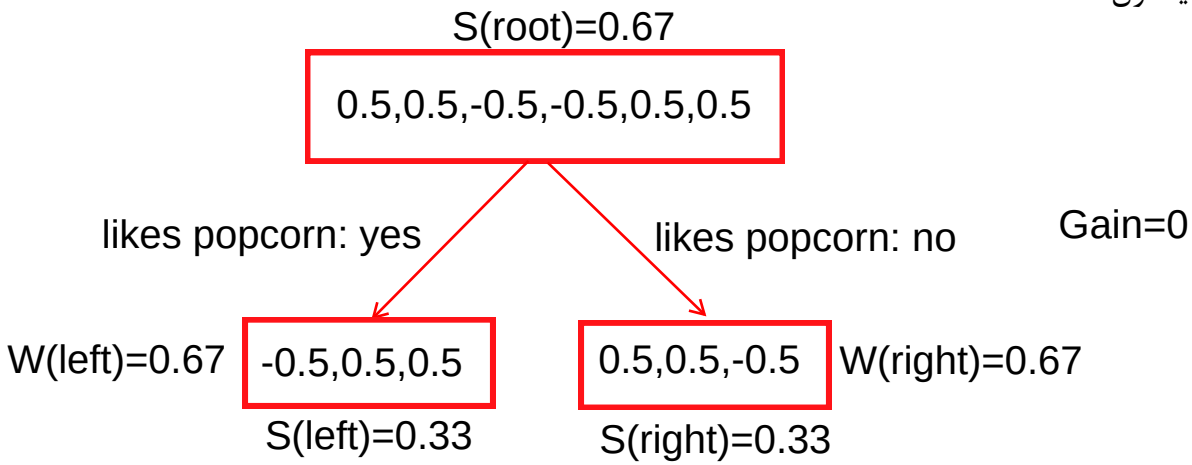
$$Gain = S_{left} + S_{right} - S_{root}$$

لذا داریم:

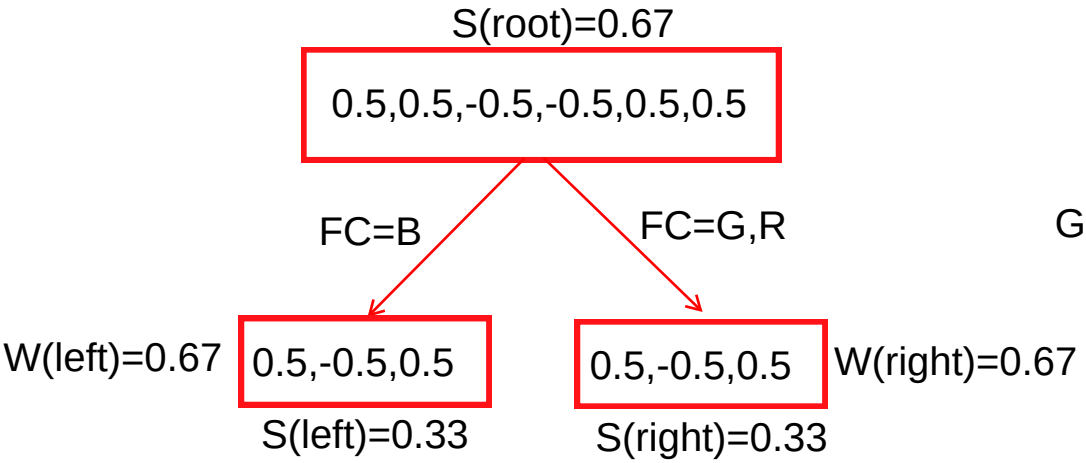
$$S_{root} = 0.67\,, S_{left} = 0.33\,, S_{right} = 0.33\,, Gain = 0$$

$$W_{left} = 0.66\,, W_{right} = 0.6$$

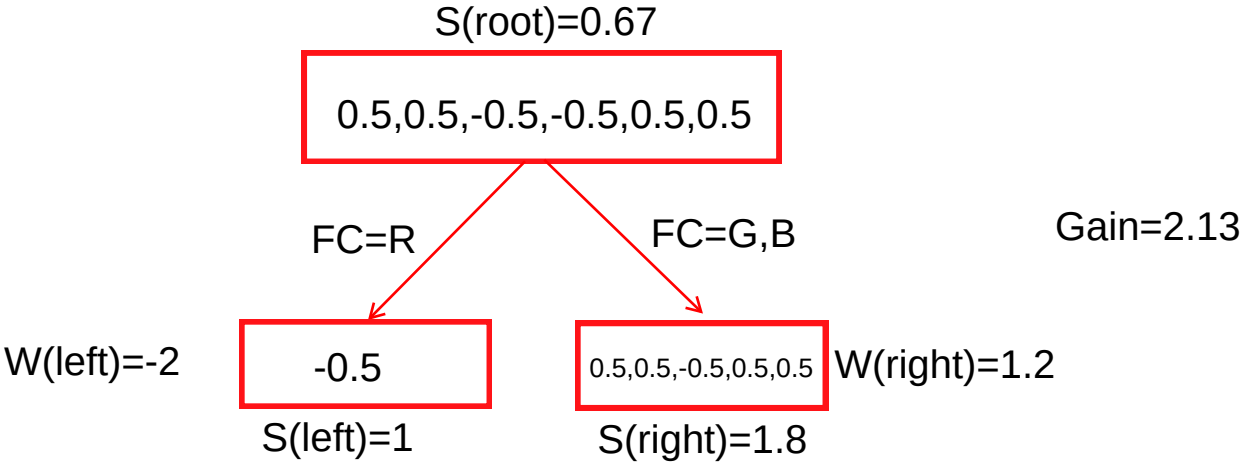
توجه شود این محاسبات را برای کاندید اول نوشتم و برای کاندید های دیگر نمینویسم و مستقیم درخت مروبطه را نمایش میدهم. کاندید اول:



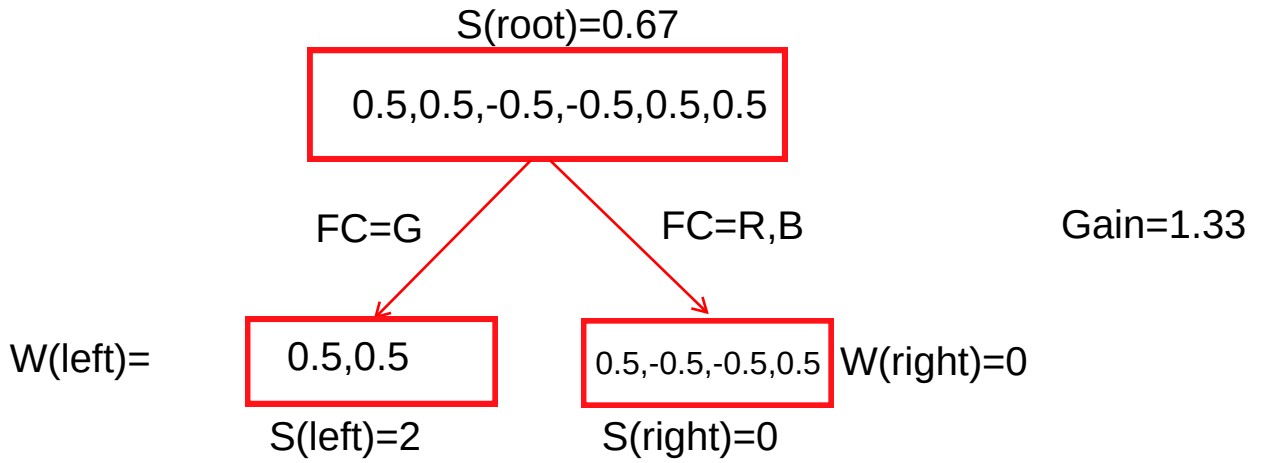
کاندید دوم:



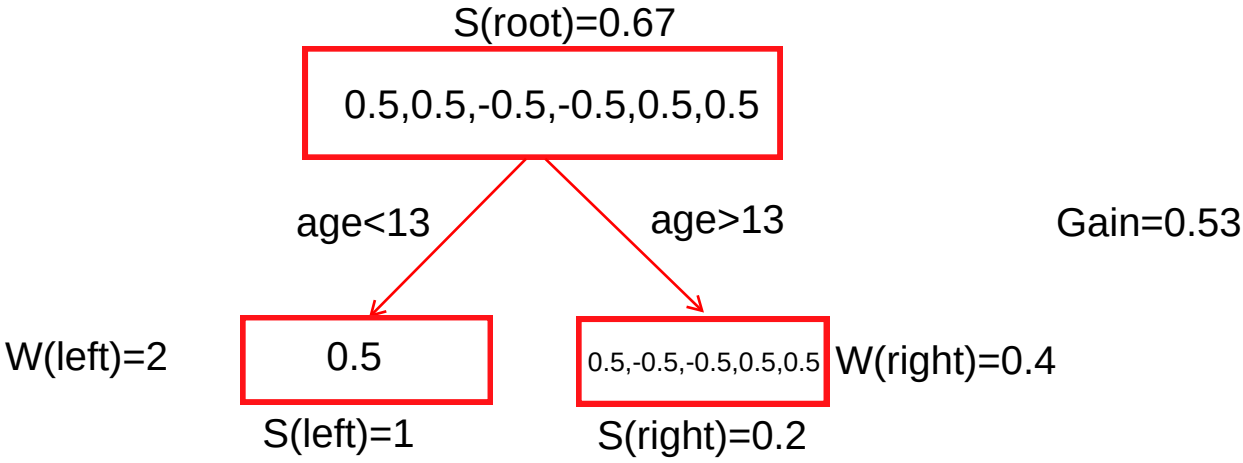
کاندید سوم:



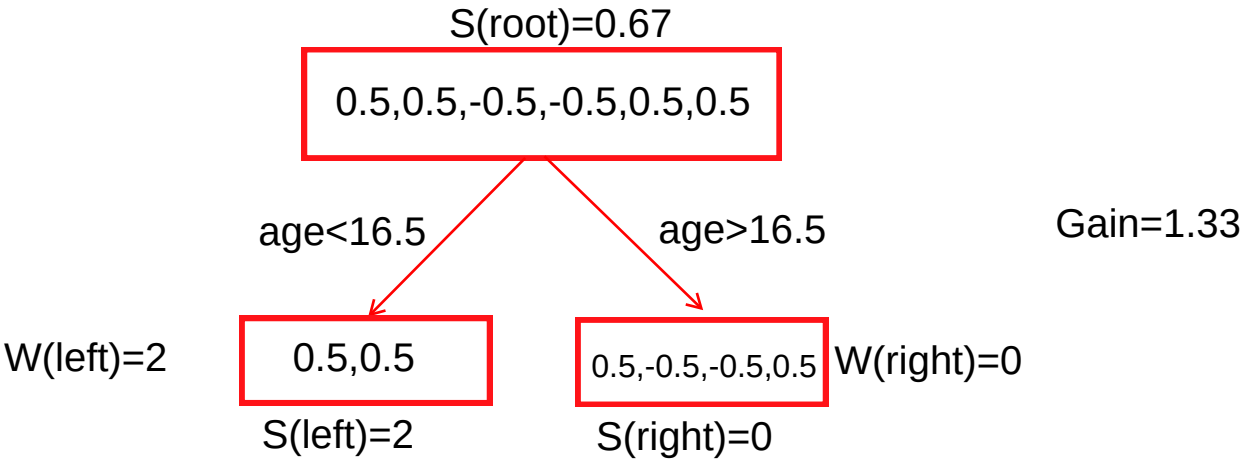
کاندید چهارم:



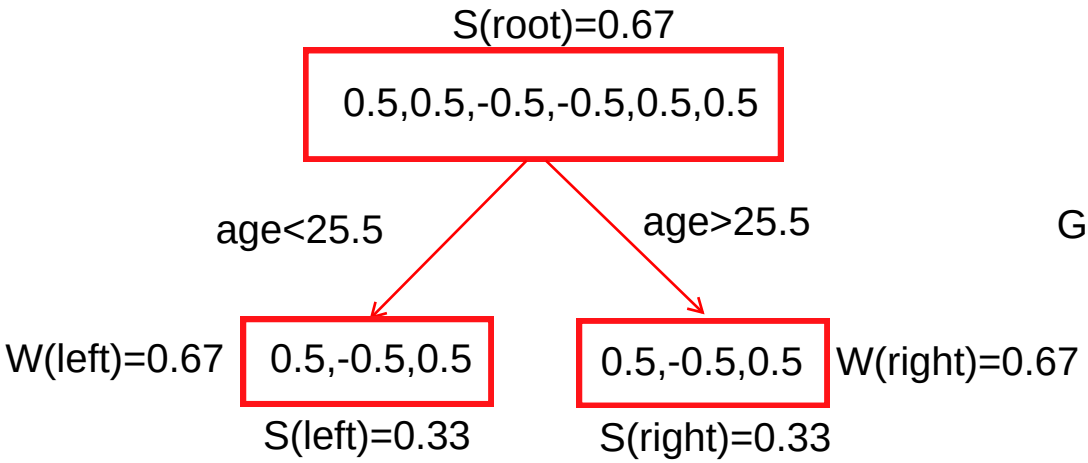
کاندید پنجم:



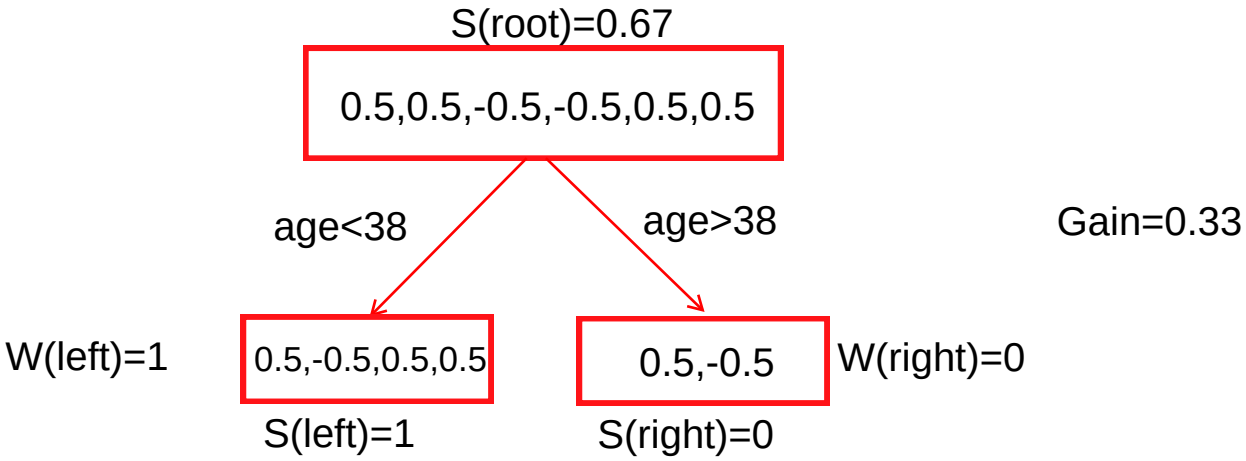
کاندید ششم:



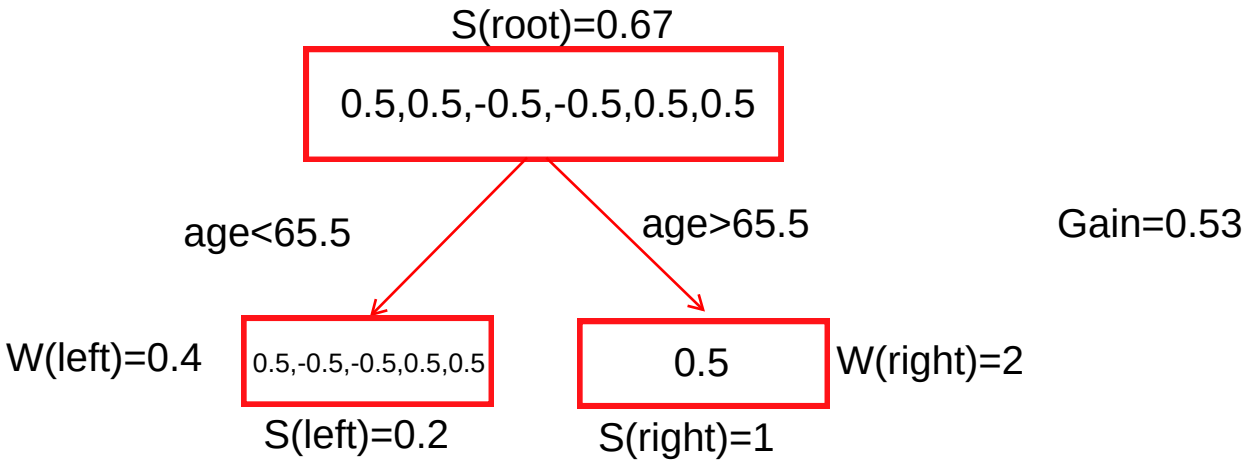
کاندید هفتم:



کاندید هشتم:



کاندید نهم:



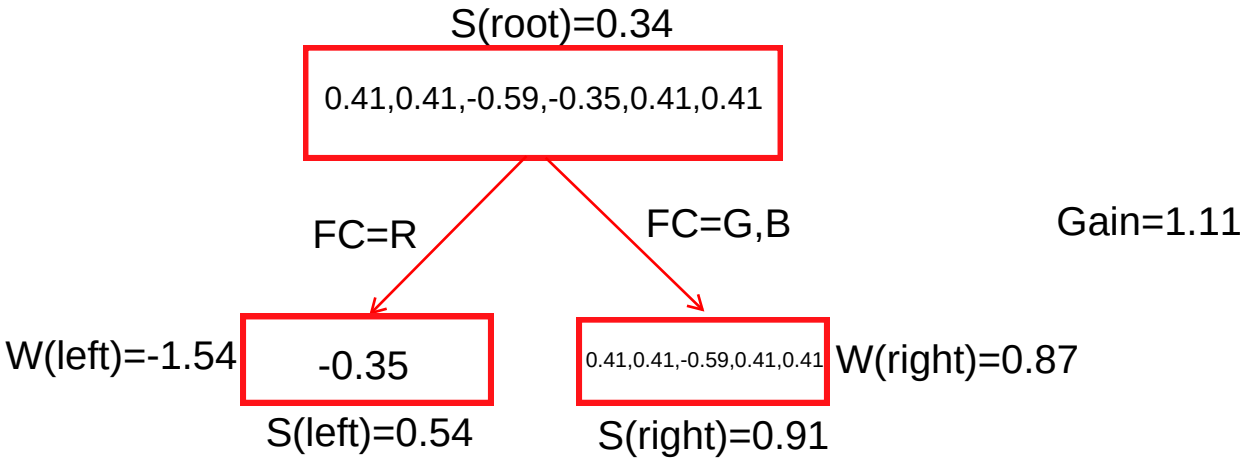
لذا بیشترین ***gain*** را کاندید سوم دارد با عدد ۲.۱۳ لذا این درخت انتخاب میشود.حال اگر گاما را ۰.۳ در نظر بگیریم،

$$\ln(odds)_1 = \ln\left(\frac{0.5}{1-0.5}\right) + 0.3 * 1.2 = 0.36\,, p_1 = sig(0.36) = 0.59$$

$$R_1 = 1 - 0.59 = 0.41$$

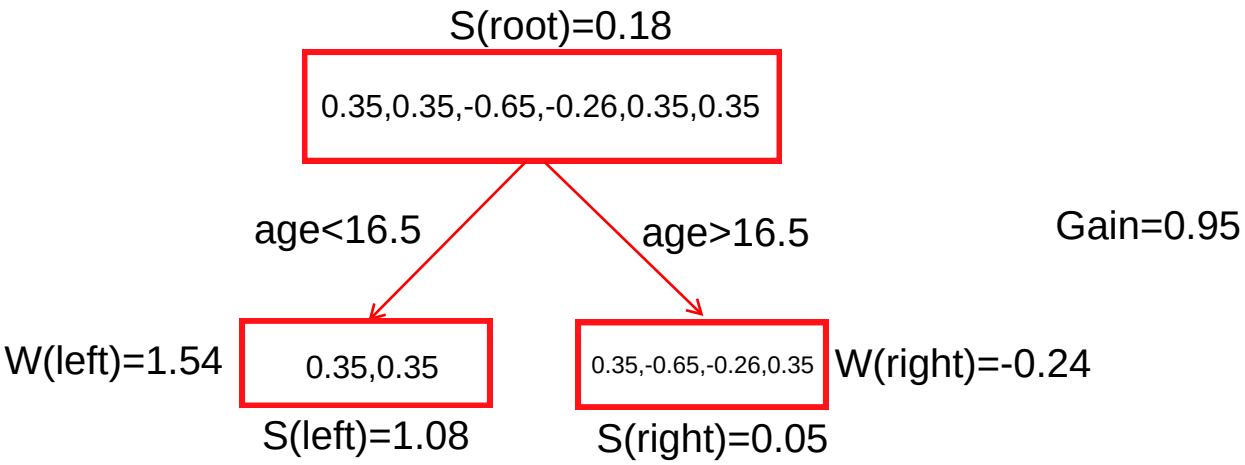
Likes popcorn	Age	Favorite color	Loves Troll 2	P1	R1
Yes	12	Blue	Yes	0.59	0.41
Yes	87	Green	Yes	0.59	0.41
No	44	Blue	No	0.59	-0.59
Yes	19	Red	no	0.35	-0.35
No	32	Green	Yes	0.59	0.41
No	14	Blue	Yes	0.59	0.41

به همین منوال درخت بعدی را انتخاب میکنیم که بخاطر زیاد بودن آن، دیگر درخت هارا هم رسم نمیکنم و درخت انتخاب شده نهایی را اینجا می آورم:



Likes popcorn	Age	Favorite color	Loves Troll 2	P1	R1
Yes	12	Blue	Yes	0.65	0.35
Yes	87	Green	Yes	0.65	0.35
No	44	Blue	No	0.65	-0.65
Yes	19	Red	no	0.26	-0.26
No	32	Green	Yes	0.65	0.35
No	14	Blue	Yes	0.65	0.35

همین طور درخت سوم:



Likes popcorn	Age	Favorite color	Loves Troll 2	P1	R1
Yes	12	Blue	Yes	0.75	0.25
Yes	87	Green	Yes	0.63	0.37
No	44	Blue	No	0.63	-0.63
Yes	19	Red	no	0.24	-0.24
No	32	Green	Yes	0.63	0.37
No	14	Blue	Yes	0.75	0.25

این مراحل باید ادامه پیدا کند که بخاطر کمبود وقت، ادامه نمیدهیم:

$$P_i = sig(0 + 0.3 * W_1 + 0.3 * W_2 + 0.3 * W_3)$$

Likes popcorn	Age	Favorite color	W0	W1	W2	W3	Pi	Loves Troll 2	Predict
Yes	12	Blue	0	1.2	0.87	1.54	0.75	1	1
Yes	87	Green	0	1.2	0.87	-0.24	0.63	1	1
No	44	Blue	0	1.2	0.87	-0.24	0.63	0	1
Yes	19	Red	0	-2	-1.54	-0.24	0.24	0	0
No	32	Green	0	1.2	0.87	-0.24	0.63	1	1
No	14	Blue	0	1.2	0.87	1.54	0.75	1	1

سوال ۴ – در نوتبوک آورده شده است.