Նախաբան

Մի քանի դասակարգման ալգորիթմների միջոցով, լուծել ենք խնդիր թե արդյոք բանկի հաճախորդը կհամաձայնի արդյոք ժամկետային ավանդ ներդնել բանկում, թե ոչ և համեմապել ենք այդ մեթոդները:

blablabla

Ներածություն

Ունենք բանկի հաճաղորդների որոշակի տվյալներ, այդ տվյալները հավաքվել են բանկի մարկետինգային բաժինների կողմից հեռախոսազանգերի միջոցով : Հավաքվել են ամեն հաճախորդի համար հետևյալ տվյալները: Նշենք նաև այն փոփոխականները որոնց վերագրված են այդ տվյալները:

Տվյալներ որոնք կապված են հաճախորդների հետ`

- *Age* -Տարիք
- *Job* -Աշխատանքի տեսակը
- *Marital* - Կարգավիճակը
- *Education* - Կրթությունը
- *Default* - Չվճարված վարկ ունի թե ոչ
- *Housing* - Հիպոթեքային վարկ ունի թե ոչ
- *Loan* – Վարկ ունի թե ոչ

Տվյալներ կապված վերջին հեռախոսազանգի հետ`

- *Contact* - Կապի միջոցը
- *Month* - Վերջին հեռախոսազանգի ամիսը
- *Dayofweek* - Վերջին հեռախոսազանգի շաբաթվա օրը
- *Duration* - Վերջին հեռախոսազանգի տևողությունը

Սոցիալական և տնտեսական տվյալներ`

- *Emp.var.rate* – Գործազրկության մակարդակը (Եռամսյակային ցուցանիշ)
- *Cons.price.idx* – Սպառողական զամբյուղի գների մակարդակի փոփոխությունը (Ամսեկան ցուցանիշ)
- *Cons.conf.idx* – Ցույց է տալիս տնտեսությունում ինչպիսին է սպառման և խնայողությունների մակարդակը (Ամսեկան ցուցանիշ)
- *Euribor3m* – 3 Ամսեկան *euribor* -ի տոկոսադրույքը
- *Nr.employed* – Աշխատանք ունեցող անձանց քանակը (Եռամսյակային ցուցանիշ)

Այլ տվյալներ`

- *Campaign*- քանի զանգ է կատարվել այս մարկետինգային արշավի ժամանակ տվյալ հաճախորդին
- *Pdays*- Քանի օր է անցել հաճախորդի հետ վերջին հեռախոսազանգից, որը կատարվել է անցած մարկենտինգային արշավի ժամանակ
- *Previous*- Նախկինում քանի հեռախոսազանգ է եղել
- *Poutcome* – Նախկին մարկենտինգային արշավի արդյունքը

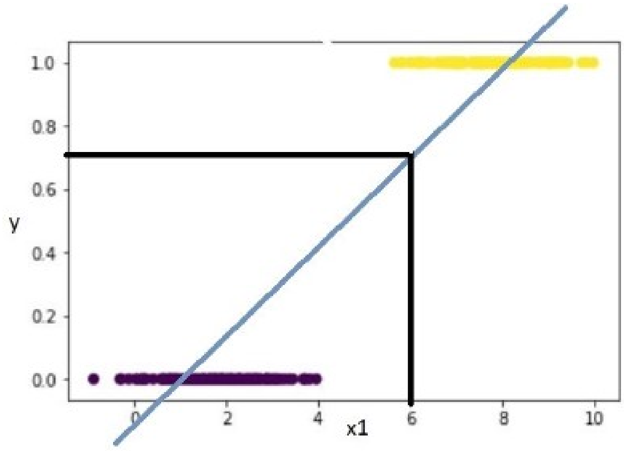
Y- Արդյո՞ք հաճախորդը համաձայնվել է ժամկետային ավանդ ներդնել բանկում, թե ոչ:

Խնդիրը կայանում է նրանում որ ունենալով 41188 հաճախորդների փվյալներ՝ պետք է $Linear\ discriminant$, $Logistic\ regression$, $Naive\ Bayes\ classifier$, $Perceptron$, SVM, $Decision\ Trees$ դասակարգման ալգորիթմները կիրառել և համեմապել դրանց արդյունքները որոշակի մեթոդներով: Լավագույն մեթոդները ընփրելով բանկը հնարավորություն կսփանա մեծ հավանականությամբ կանխապեսել, թե որ հաճախորդները կհամաձայնվեն ներդնել ավանդ, հետևաբար կզանգահարի միայն նրանց և կխնայի գումար և ժամանակ:

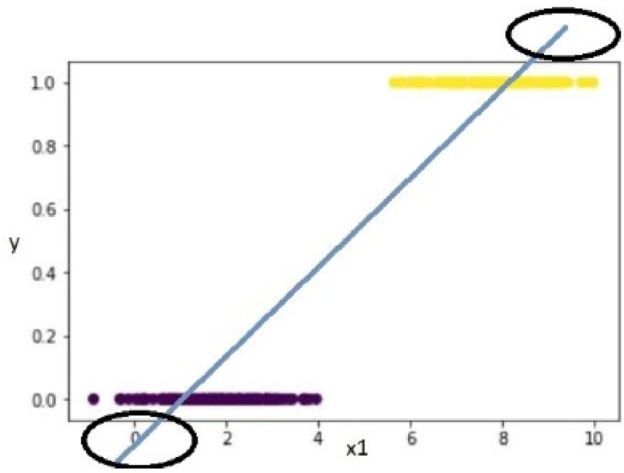
Խնդրի լուծման համար նախ ներկայացվել են ամեն ալգորիթմի էությունը և մաթեմապիկական մոդելները: Այնուհետև ներկայացվել են ալգորիթմների որակը սփուգելու մի քանի մեթոդներ: $Python$ ծրագրավորման լեզվի միջոցով կիրառել ենք ալգորիթմները և համեմապել դրանք:

Logistic regression

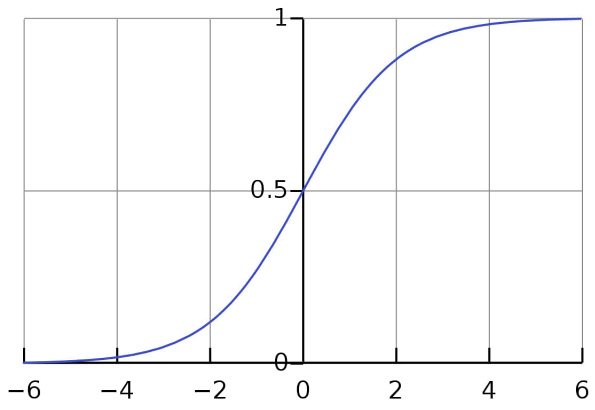
Logistic regression մեթոդը իր մեջ ներառում է գծային ռեգրեսիայի ալգորիթմը: Այս դեպքում մենք առնչվում ենք բինար դասակարգման հետ, այսինքն փորձում ենք պարզել թե մեր փվյալը երկու դասերից որին է պապկանում:
Սպորև պեղադրված նկարում կիրառելով գծային ռեգրեսիա կսպանանք հետևյալ ուղիղը:



Օրինակ եթե ունենք փվյալ, որի համար $x_1 = 6$ կսպանանք, որ $y = 0.7$, որը ավելի մոտ է 1-ին քան 0-ին: Սա լավ է, բայց եթե մեր փվյալի x_1 արժեքը մեծ լինի 1ից կամ փոքր 0-ից մենք չենք կարողանա ոչ մի ենթադրություն անել:



Մեզ անհրաժեշտ է *sigmoid* ֆունկցիան կամ այլ կերպ ասած *Logistic function*-ը, որը ընդունում է արժեքներ $[0, 1]$ միջակայքում:



X մատրիցը իր մեջ ներառում է բոլոր տվյալները: $X \in R^{m\times n}$, որտեղ m-ը մեր ունեցած տվյալների քանակն է, իսկ n-ը՝ տվյալների բնութագրիչների քանակը: y-ը մեր արդյունքների վեկտորն է, այսինքն ցանկացած տվյալի համար $y = 1$, թե $y = 0$: Մեր խնդիրն է՝ ունենալով X մատրիցը կանխատեսել հավանականությունը, որ $y = 1$: Քանի որ մենք ցանկանում ենք գտնել հավանականություն, ապա մեր մոդելում արդյունքները պետք է ընդունեն արժեքներ $[0, 1]$ միջակայքից: Վերցնենք $P(x) = \frac{1}{1+e^{-x}}$ ֆունկցիան: Որպեսզի այս հավանականությունը համապատասխանի մեր ունեցած տվյալներին անհրաժեշտ է կապարել որոշակի ձևափոխություններ գծային ռեգրեսիայի նման: Սահմանենք պարամետրերի վեկտոր $w \in R^n$, բնութագրիչները պարամետրիզացված կունենան հետևյալ տեսքը՝

$$w_1x_1 + w_2x_2 + ... + w_nx_n = \sum_{i=1}^{n} w_ix_i = w^TX \tag{1}$$

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}^T \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = [w_1, w_2, ..., w_n] \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = w_1x_1 + w_2x_2 + ... + w_nx_n$$

$$P(w^Tx) = \frac{1}{1 + e^{-w^Tx}}$$

Եթե (1)-ը ոչ բացասական է, ապա կարող ենք ասել որ $y = 1$, քանի որ $P(x) \geq 0.5$: Եթե մեր կշիռները(գործակիցները) օպտիմալ որոշված լինեն, ապա կարող ենք ճիշտ կանխատեսել $y = 1$ լինելու հավանականությունը: Կարող ենք այս հավանականությունը գրել պայմանական հավանականության լեզվով

$$P(y = 1|x; w) = \frac{1}{1 + e^{-w^Tx}}$$

որտեղ $x; w$-ն նշանակում է, որ X-ը պարամետրիզացվել է w-ի միջոցով:

Միակ խնդիրը այն է, որ մենք չենք կարող լոգիստիկ ֆունկցիան տեղաշարժել, սակայն մենք կարող ենք այն ձգել: Բայց դրա համար մեզ հարկավոր է w_0, որը կախված չէ բնութագրիչներից: Դրա համար X-ի մեջ ավելացնում ենք 1-երի փող և w-ի մեջ w_0: Մեր X-ի և w-ի երկարությունները դառնում են $n+1$:

$$P(y=1|x;w) = \frac{1}{1+e^{-w^T x}} \Rightarrow e^{w^T x} = \frac{P(y=1|x;w)}{1-P(y=1|x;w)} \Rightarrow \ln \frac{P(y=1|x;w)}{P(y=0|x;w)} = w^T x$$

$$\frac{P(y=1|x;w)}{P(y=0|x;w)}$$

սա կոչվում է $odds\ ratio$, որը ցույց է տալիս, թե քանի անգամ է $P(y=1|x;w)$ մեծ $P(y=0|x;w)$-ից:

$$ln\frac{P(y=1|x;w)}{P(y=0|x;w)}$$

սա կոչվում է $log-odds\ ratio$

Յուրաքանչյուր գործակից կարևորության աստիճան է տալիս բնութագրիչներին: Հարց է առաջանում, թե ինչպես գտնել ճիշտ գործակիցները: Գրենք հետևյալը՝

$$P(Y=y) = p^y(1-p)^{1-y} = \begin{cases} p, & \text{եթե } y=1 \\ 1-p, & \text{եթե } y=0 \end{cases} \tag{2}$$

որտեղ p-ն մեր գնահատված հավանականությունն է: Մեզ անհրաժեշտ է, որ P-ն լինի մեծագույնը: Եթե գրենք յուրաքանչյուր փվյալի համար առանձին և բազմապատկենք իրար կստանանք L ֆունկցիան: Սա ամեն մի կանխատեսման համար է: Որպեսզի բոլոր կանխատեսումները նայենք միասին, անհրաժեշտ է բոլորը բազմապատկել իրար՝

$$L(w|y) = \prod_{i=1}^{m} P(Y=y_i) = \prod_{i=1}^{m} P(y=1|x;w)_i^{y_i}(1-P(y=1|x;w_i)_i)^{1-y_i}$$

Մեր նպատակն է գտնել այն w-ն, որը կմաքսիմալացնի $L(w|y)$-ը, որի շնորհիվ $P(y=1|x;w)$ կնկարագրի մեր փվյալները ավելի լավ: Գտնելով w-ն և տեղադրելով $sigmoid$ ֆունկցիայի մեջ կստանանք լավագույն արդյունքը:

Linear Discriminant

Ենթադրենք ունենք նմուշ C_1 և C_2 դասերից: Մեր նպատակն է գտնել մի վեկտոր, որի վրա տվյալները պրոյեկտելիս դրանք լավագույն կերպով կարող են առանձնացվել 2 դասերի միջև:

$$z = \omega^T x$$

Սա x-ի պրոյեկցիան է ω-ի վրա, որը d չափանի տարածությունը արտապատկերում է 1 չափանի տարածության վրա: x-ը նմուշների վեկտորն է:

$$x^t \in R^d$$

$m_1 \in R^d$ - C_1 դասի նմուշների միջինն է մինչև պրոյեկտելը:
$\boldsymbol{m_1} \in R$ - C_1 դասի նմուշների միջինն է պրոյեկտելուց հետո:
m_2 և $\boldsymbol{m_2}$ համապատասխանաբար C_2 դասի համար:
Մեզ տրված նմուշների բազմություն՝ $X = \{x^t, r^t\}$ $t \in 1, ..., n$ որտեղ n-ը մեր ունեցած տվյալների քանակն է:

$$r^t = \begin{cases} 1, & \text{եթե } x^t \in C_1 \\ 0, & \text{եթե } x^t \in C_2 \end{cases} \tag{3}$$

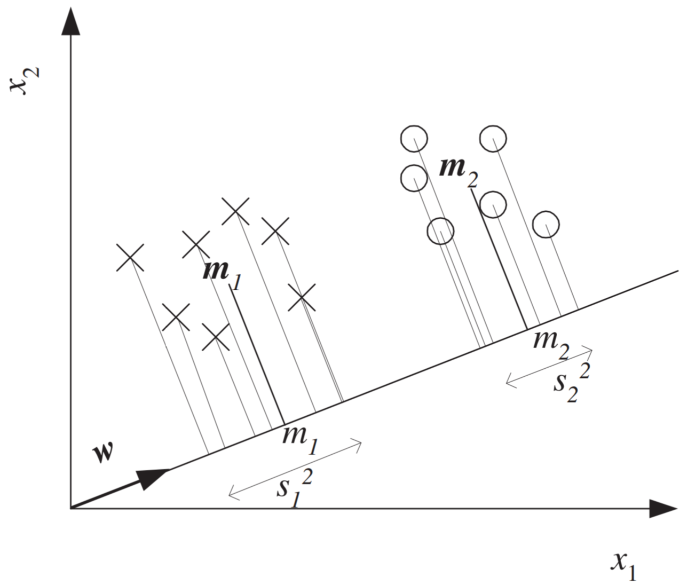
$$\boldsymbol{m_1} = \frac{\sum_t \omega^T x^t r^t}{\sum_t r^t} = \omega^t m_1$$

$$\boldsymbol{m_2} = \frac{\sum_t \omega^T x^t (1-r)^t}{\sum_t (1-r)^t} = \omega^t m_2$$

$$\boldsymbol{S_1^2} = \sum_t (\omega^T x^t - m_1)^2 r^t$$

$$\boldsymbol{S_2^2} = \sum_t (\omega^T x^t - m_2)^2 (1-r)^t$$

S_1^2 -ն C_1 դասի նմուշների ցրվածքն է պրոյեկտելուց հետո: S_2^2 -ն համապատասխանաբար C_2 դասի:



Պրոյեկտելուց հետո մենք ցանկանում ենք,որ միջինները հնարավորինս հեռու լինեն իրարից իսկ նույն դասի նմուշները հնարավորինս մոտ լինեն իրար: Այսինքն $|\boldsymbol{m_1}-\boldsymbol{m_2}|$-ը լինի հնարավորինս մեծ,իսկ $S_1^2+S_2^2$ -ը՝ փոքր: Ֆիշերի $Linear Discriminant$ -ը այն ω-ն է,որը մաքսիմալացնում է հետևյալ ֆունկցիան՝

$$v(\omega)=\frac{\boldsymbol{(m_1-m_2)^2}}{\boldsymbol{S_1^2+S_2^2}}$$

$$\boldsymbol{(m_1-m_2)^2}=(\omega^T m_1-\omega^T m_2)^2=\omega^T(m_1-m_2)(m_1-m_2)^T\omega=\omega^T S_B\omega$$

որտեղ՝

$$S_B=(m_1-m_2)(m_1-m_2)^T$$

$\boldsymbol{S_1^2}=\sum_t(\omega^T x^t-m_1)^2 r^t=\sum_t \omega^T(x^t-m_1)(x^t-m_1)^T\omega r^t=\omega^T S_1\omega$
$S_1=\sum_t r^t(x^t-m_1)(x^t-m_1)^T$
$S_2=\sum_t(1-r)^t(x^t-m_2)(x^t-m_2)^T$
$\boldsymbol{S_1^2+S_2^2}=\omega^T S_1\omega+\omega^t S_2\omega=\omega^T S_\omega\omega$
$S_w=S_1+S_2$

$$v(\omega)=\frac{\omega^T S_B\omega}{\omega^T S_\omega\omega}=\frac{|\omega^T(m_1-m_2)|^2}{\omega^T S_\omega\omega}$$

Ածանցենք և հավասարեցնենք 0-ի:
$2\frac{\omega^T(m_1-m_2)}{\omega^T S_\omega\omega}((m_1-m_2)-\frac{\omega^T(m_1-m_2)}{\omega^T S_\omega\omega}S_\omega\omega)=0$
$\frac{\omega^T(m_1-m_2)}{\omega^T S_\omega\omega}$-հաստատուն է, նշանակում ենք C-ով:
$(m_1-m_2)-CS_\omega\omega=0\Rightarrow\omega=CS_\omega^{-1}(m_1-m_2)$ քանի որ մեզ հետաքրքիր է ուղղությունը կարող ենք վերցնել $C=1$ և գտնել ω-ն:

$Naive\ Bayes$

Մեքենայական ուսուցման մեջ $Naive\ Bayes$ -ը պատկանում է հավանակային դասակարգիչների ընտանիքին: Այն հիմնվում է Բայեսյան թեորեմի վրա, որտեղ ենթադրվում է անկախություն բնութագրիչների միջև: Այս մոտեցումը հիմք է առել 1960-ականներից: Այն լայնորեն կիրառվում է տեքստային դասակարգման, փաստաթղթերի տեսակավորման նաև բժշկական հետազոտությունների համար: Ենթադրենք մենք ցանկանում ենք դասակարգել արդյոք վարկառուն ունի բարձր ռիսկայնություն թե ցածր: Դիտարկենք վարկառուի պարեկան խնայողությունները և եկամուտները որոնք ներկայացվում են X_1 և X_2 պատահական մեծություններով: Վարկառուի վարկունակությունը ներկայացնենք Բեռնուլի պատահական մեծության միջոցով: Եթե $C = 1$, ապա վարկառուն դասակարգվում է որպես ռիսկային հաճախորդ, իսկ եթե $C = 0$, ապա որպես ոչ ռիսկային: Եթե մենք գիտենք հետևյալ հավանականությունը $P(C = 1|X_1, X_2)$, և նոր դիմում ենք ստանում $X_1 = x_1$, $X_2 = x_2$ տվյալներով ,ապա $C = 1$ եթե $P(C = 1|x_1, x_2) > 0.5$ և $C = 0$ հակառակ դեպքում: Կամ նույն է, որ գրենք

$$C = \begin{cases} 1, \text{ եթե } P(C=1|x_1,x_2) > P(C=0|x_1,x_2) \\ 0, \text{ հակառակ դեպքում} \end{cases} \tag{4}$$

Սխալի հավանականությունը կլինի

$$1 - \max(P(C = 1|x_1, x_2), P(C = 0|x_1, x_2))$$

Խնդիրը կայանում է նրանում որ պետք է հաշվենք $P(C|x)$ որտեղ $x = [x_1, x_2]$: Օգտվելով Բայեսի կանոնից կարող ենք գրել

$$P(C|x) = \frac{P(C)p(x|C)}{p(x)}$$

$P(C = 1)$ ցույց է տալիս հավանականություն որ հաճախորդը քիչ ռիսկային է:

$$P(C = 1) + P(C = 0) = 1$$

$p(x|C)$ մեր դեպքում $p(x_1, x_2|C = 1)$ցույց է տալիս հավանականություն որ $X_1 = x_1$, $X_2 = x_2$ եթե գիտենք, որ հաճախորդը ռիսկային է:

$$p(x) = \sum_C p(x, C) = p(x|C = 1)P(C = 1) + p(x|C = 0)P(C = 0)$$

ցույց է տալիս հավանականություն որ $X_1 = x_1$, $X_2 = x_2$:

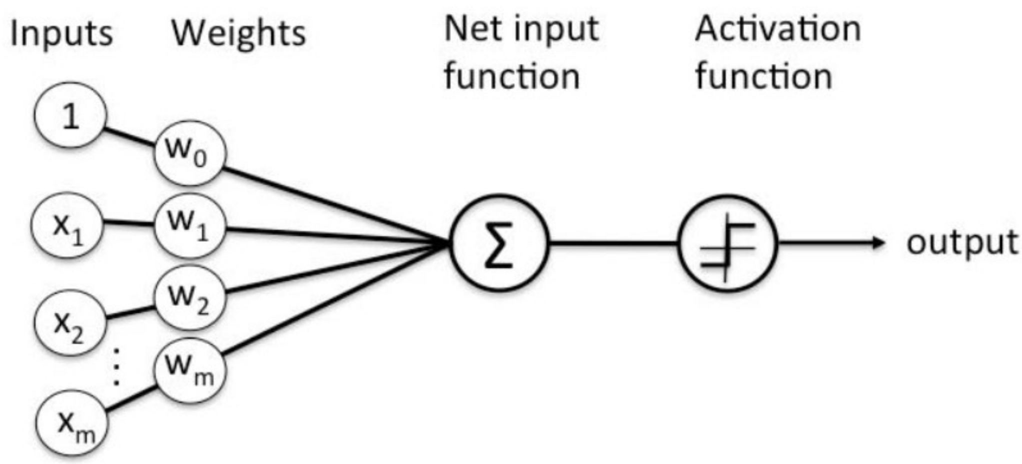
$Perceptron$

x-ներմուծված տվյալի արժեքների վեկտորն է:

ω-կշիռներից կազմված վեկտորն է:

ω_0-փոփոխական է, որը հնարավորություն է տալիս կարգավորել ֆունկցիան մեր նախընտրած ձևով, որը կախված չէ ներմուծված արժեքներից:

$Perceptron$-ը ստանում է արժեքները, բազմապատկում դրանք որոշակի կշիռներով, կատարում է որոշակի ձևափոխություն և վերադարձնում է արդյունք որոշակի ֆունկցիայի միջոցով:

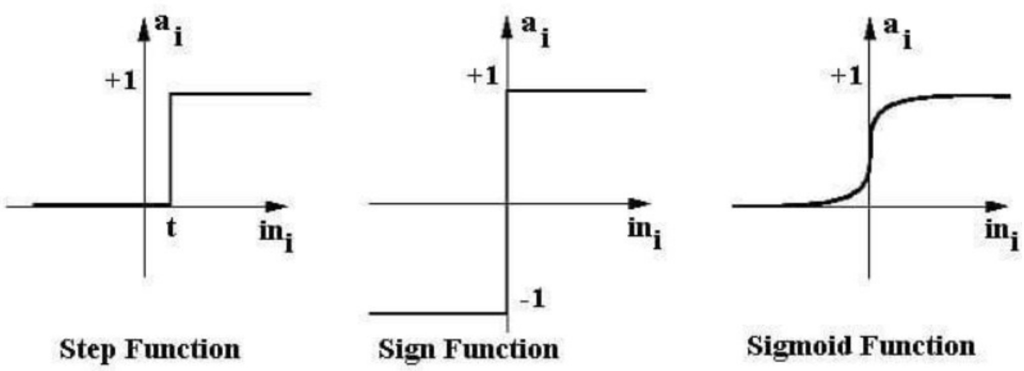


որտեղ՝

$$\sum = \omega_0 + \omega_1 x_1 + ... + \omega_n x_n$$

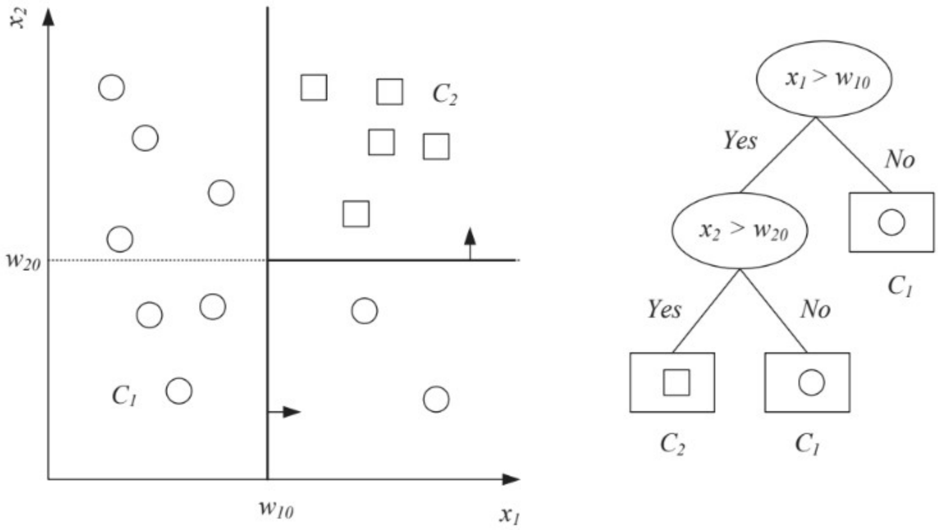
Կարող են լինել տարբեր ակտիվացիայի ֆունկցիաներ, օրինակ՝

$$f(x) = \begin{cases} 1, & \text{եթե } \omega_0 + \omega x > 0 \\ 0, & \text{հակառակ դեպքում} \end{cases} \tag{5}$$



Decision trees

Այս ալգորիթմի ժամանակ հարկավոր է ներմուծված տվյալները դասակարգել որոշակի դասերի միջև: Այստեղ դա տեղի է ունենում հետևյալ կերպ: Ընդունելով տվյալները մենք այն բաժանում ենք տարբեր մասերի որոշակի ֆունկցիայով: Այնուհետև նույն գործողությունը կատարում ենք արդեն բաժանված մասերի համար: Այսպես շարունակում ենք այնքան ժամանակ մինչև մեր տվյալը համապատասխանի որոշակի դասի: Ծառերը կազմված են արմատից, տերևներից և հանգույցներից: Յուրաքանչյուր հանգույցին համապատասխանում է որոշակի թեստային ֆունկցիա: Վերցնելով ներմուծված տվյալները ամեն հանգույցում կատարվում է փորձ և անցում է կատարում հաջորդ հանգույցին: այսպես այնքան ժամանակ մինչև հասնենք տերևին, որին համապատասխանում է որոշակի դաս:



Տերևները նույնպես հանգույցներ են: Յուրաքանչյուր հանգույցին հասնելու համար անհրաժեշտ է որոշակի ֆունկցիա: m-րդ հանգույցին համապատասխանում է $f_m(x)$ ֆունկցիան: Հանգույցները միացնող հատվածը կոչվում է ճյուղ: Այս մեթոդը մեզ թույլ է տալիս արագ դասակարգել տվյալները:

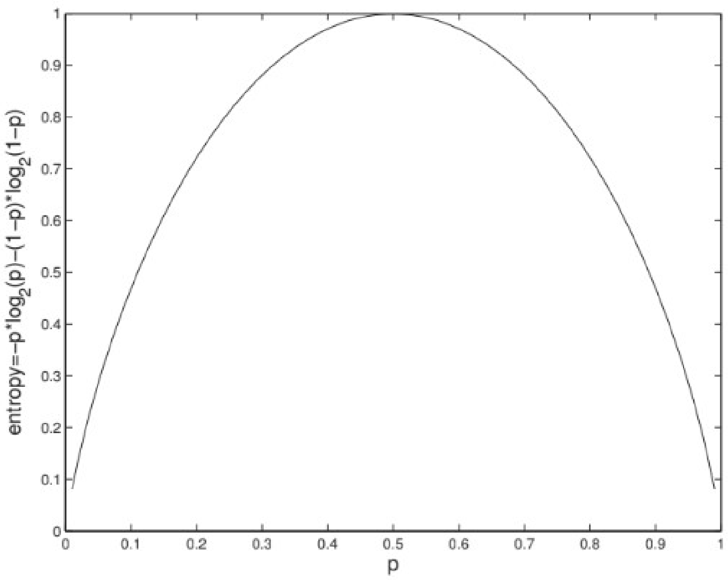
Այս ալգորիթմի ժամանակ մենք միշտ ցանկանում ենք հնարավորինս լավ կիսել մեր տվյալները: Կան մի քանի մեթոդներ ծառերը ստեղծելու համար, որոնցից մեկը *CART algorithm*-ն է:

$CART\ algorithm$

Այս ալգորիթմի ժամանակ մենք ցանկանում ենք ճիշտ հարց տալ ճիշտ ժամանակին, որպեսզի մեր ծառը հնարավորինս կարճ ստացվի: Բայց ինչպեն հասկանանք, թե որն է լավագույն հարցը կամ որն է ճիշտ դիրքը: Այս հարցերին պատասխանելու համար հարկավոր է հաշվել $Information\ gain$-ը: Որպեսզի հասկանանք, թե ինչ է $Information\ gain$-ը նախ պետք է ներկայացնենք թե ինչ է $entropy$-ն: $Entropy$-ն ֆունկցիա է, որը չափում է անճշտությունը: Տեսնենք ինչ ենք հասկանում անճշտություն ասելով: Ենթադրենք ունենք խնձորներով լի զամբյուղ և մեկ այլ զամբյուղ լի թղթերով, որոնց վրա գրված է խնձոր: Եվ եթե ամեն զամբյուղից վերցնենք մեկական առարկա, ապա հավանականությունը, որ գրվածը կհամապատասխանի վերցված առարկային 1 է, իսկ անճշտությունը կլինի 0: Բայց եթե ունենք 4 տարբեր մրգեր և 4 թղթեր համապատասխանաբար մրգերի անունները վրան գրված, ապա պարզ է որ մեկական առարկա վերցնելու դեպքում համապատասխանելու հավանականությունը փոքր կլինի 1-ից, իսկ անճշտությունը այլևս 0 չի լինի: Վերադառնանք $entropy$-ին: Ասացինք որ այն չափում է անճշտությունը: Դիտարկենք 2 դասանի խնդիր, որի մեջ պետք է կայացնենք որոշում "այո" կամ "ոչ"-ի միջև: "այո"-ի հավանականությունը p է, իսկ "ոչ"-ինը՝ $1-p$: Այս դեպքում $entropy$-ն հավասար կլինի հետևյալին՝

$$\Phi(p, 1-p) = -p \log_2 p - (1-p)\log_2(1-p)$$

Եթե սա ներկայացնենք գրաֆիկորեն ապա կտեսնենք, որ երբ $p = 0$ կամ $p = 1$ Φ-ն հավասար է 0-ի և հավասար է 1-ի երբ $p = 0.5$, քանի որ այս դեպքում անճշտությունը ամենաբարձրն է:



Սակայն բացի $entropy$-ն կան նաև որիշ ֆունկցիաներ, որոնցով կարող ենք հաշվել անճշտությունը: Օրինակ ոչ բացասական ֆունկցիաները, որոնք բավարարում են հետևյալ հատկություններին՝

1. $\Phi(\frac{1}{2},\frac{1}{2}) \geq \Phi(p,1-p)\ \forall p \in [0,1]$
2. $\Phi(0,1) = \Phi(1,0) = 0$
3. $\Phi(p,1-p)$-ն աճում է $[0,\frac{1}{2}]$-ում և նվազում $[\frac{1}{2},1]$-ում ըստ p-ի:

Այդպիսի ֆունկցիաներից են՝

1. $\Phi(p,1-p) = -p\log_2 p - (1-p)\log_2(1-p)$- $Entropy$
2. $\Phi(p,1-p) = 2p(1-p)$- $Gini\ index$
3. $\Phi(p,1-p) = 1 - \max(p,1-p)$- $Misclassification\ error$

Եվ փորձը ցույց է տվել, որ սրանց միջև մեծ տարբերություն չկա: Վերադառնանք $Information\ gain$-ին, որը ցույց է տալիս, թե ինչքանով է նվազում $entropy$-ն: Դիցուք N-ը մեր ամբողջ ընտրանքն է: Սկսենք կառուցել ծառը: p^1-ով նշանակենք "այո"-ի հավանականությունը, p^2-ով՝ "ոչ"-ի: p^1_m-ը m-րդ բնութագրիչի "այո"-ի հավանականությունն է, որը հավասար է $\frac{N^1_m}{N_m}$ և $p^2_m = \frac{N^2_m}{N_m}$: N^1_m-ը m-րդ բնութագրիչի մեջ "այո"-ների քանակն է: 1-ին քայլում հաշվում ենք մեր ամբողջ տվյալների համար $entropy$-ն:

$$E(N) = -p^1\log_2 p^1 - p^2\log_2 p^2$$

Այժմ պետք է ընտրենք թե մեր բնութագրիչներից որ մեկը դնենք արմատում: Դիցուք n հատ բնութագրիչ ունենք: Յուրաքանչյուրի համար պետք է հաշվենք $entropy$-ն և $Information\ gain$-ը: N_{mj}-ն ցույց է տալիս, թե N_m-ից քանիսն են պատկանում m բնութագրիչի j տարբերակին:

$$p^1_{mj} = \frac{N^1_{mj}}{N_{mj}}$$

ցույց է տալիս m բնութագրիչի j-րդ տարբերակի "այո"-ների հավանականությունը:

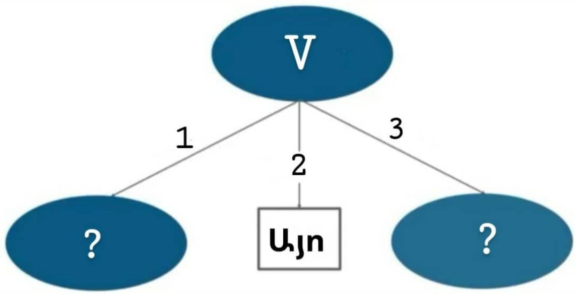
$$E(N_{mj}) = -p^1_{mj}\log_2 p^1_{mj} - p^2_{mj}\log_2 p^2_{mj}$$

$$I_m = \sum_{j=1}^{k} \frac{N_{mj}}{N_m} E(N_{mj})$$

m բնութագրիչից ստացած ինֆորմացիան

$$IG_m = E(N) - I_m$$

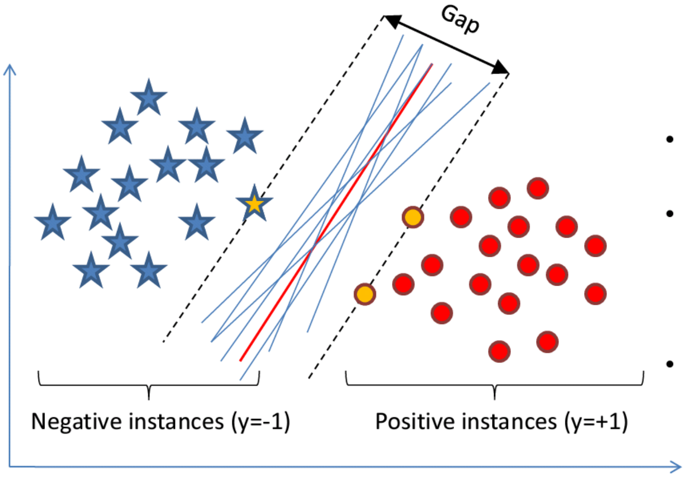
m բնութագրիչի համար $Information\ gain$-ը: Նույն կերպ բոլորի համար գտնում ենք IG-ն և ընտրում ենք ամենամեծը, որպես ծառի արմատ: Ենթադրենք ընտրել ենք V-րդ բնութագրիչը: Եվ այն ունի 3 տարբերակ: Օրինակ եթե 2-րդ տարբերակի համար բոլորը այո լինեն մենք այն կդարձնենք տերև: Այսինքն այն տվյալները, որի համար V բնութագրիչի 2-րդ տարբերակը "այո" է մենք միանգամից կդասակարգենք "այո" դասի մեջ:



Մյուս 2-ի համար այս հաշվարկները պետք է նորից կատարենք այնքան ժամանակ մինչև բոլորի համար տերևներ ստեղծվեն: Հաճախ հնարավոր է, որ մեծ ծառ առաջանա, այդ պատճառով դրվում են սահմանափակումներ: Եթե $I_m < \theta$ միանգամից տերև դարձնի:

$Support\ Vector\ Machine$

Դիցուք ունենք փվյալների բազմությունն: Մենք ցանկանում ենք բաժանել այդ փվյալները 2 դասի՝ ուղիղ գծով: Սակայն կան անվերջ քանակի ուղիղ գծեր, որոնք բաժանում են այդ դասերը: Այն փվյալները որոնք ամենամոտնն են մեր ուղղին կոչվում են $Support\ vectors$: Այդ կետերով փանենք մեր բաժանող ուղղին զուգահեռ ուղիղներ և դրանց հեռավորությունը կանվանենք Gap: SVM ալգորիթմը ընտրում է այն հիպերպլանը, որի համար Gap-ը ամենամեծն է: Այսինքն մեր դասերը հնարավորինս հեռու կզտնվեն միմյանցից:



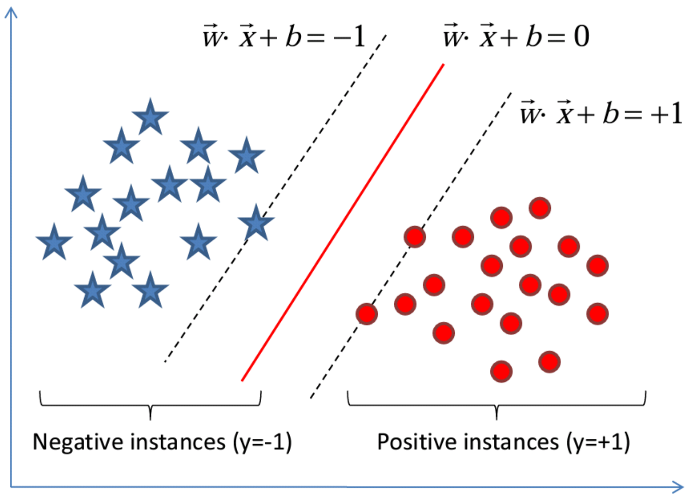
Դեպք 1: Մենք ունենք փվյալների բազմություն, որը կարող ենք առանձնացնել 2 դասերի ուղիղ գծով(եթե փվյալները R^2-ից են, եթե R^3-ից են՝ հարթությամբ): Այսինքն գծորեն առանձնացվելի փվյալներ են:

Դիցուք ունենք N փվյալ $\vec{x_1}$,$\vec{x_2}$,...,$\vec{x_N}\in R^n$

$\vec{y_1}$,$\vec{y_2}$,...,$\vec{y_N}\in(-1,1)$

y-ների արժեքները ցույց են փալիս, թե որ դասին են պատկանում մեր փվյալները: Հիպերպլանի հավասարումը փրված է հետևյալ կերպով $\vec{\omega}\vec{x}+b=0$: Պարզության համար նայենք թե 2 դասանի դեպքում ինչպես կսփացվի գրաֆիկորեն: Մենք ցանկանում ենք որպեսզի հնարավորինս մեծ լինի M-ը, բայց կախված ω-ից և b-ից կարող ենք անվերջ քանակի այդպիսի հիպերպլաններ ընտրել: Դրա համար վերցնում ենք հիպերպլանի զուգահեռ ուղիղներ՝ $\vec{\omega}\vec{x}+b=-1$ և $\vec{\omega}\vec{x}+b=1$, այս ուղիների հեռավորությունը կլինի $\frac{|b_1-b_2|}{||\vec{\omega}||}=\frac{2}{||\vec{\omega}||}$ և պետք է սա մաքսիմալացնել, այսինքն մինիմալացնել $||\vec{\omega}||$-ն կամ նույնն է, որ մինիմալացնենք $\frac{1}{2}||\vec{\omega}||^2$-ը: Մենք պետք է գործակիցները այնպես ընտրենք, որ բոլոր փվյալները ճիշտ դասակարգված լինեն, այսինքն՝

$$\begin{cases}\vec{\omega}\vec{x}+b\leq-1, \text{ եթե } y_i=-1\\ \vec{\omega}\vec{x}+b\geq 1, \text{ եթե } y_i=+1\end{cases}\tag{6}$$



Որը կարող ենք գրել հետևյալ կերպ՝

$$y_i(\vec{\omega}\vec{x}+b) \geq 1 \qquad \forall i = 1,\ldots,N$$

Այսինքն մենք ցանկանում ենք մինիմալացնել $\frac{1}{2}||\vec{\omega}||^2$-ը պայմանով, որ $y_i(\vec{\omega}\vec{x}+b) \geq 1$ $\forall i = 1,\ldots,N$: Եթե հետո նոր կվյալ սպանանք կարող ենք կանխապեսել նրա դասը այս ֆունկցիայի միջոցով՝

$$f(\vec{x}) = sign(\vec{\omega}\vec{x}+b)$$

Կազմենք Լագրանժի ֆունկցիան, որպեսզի գտնենք մինիմումը

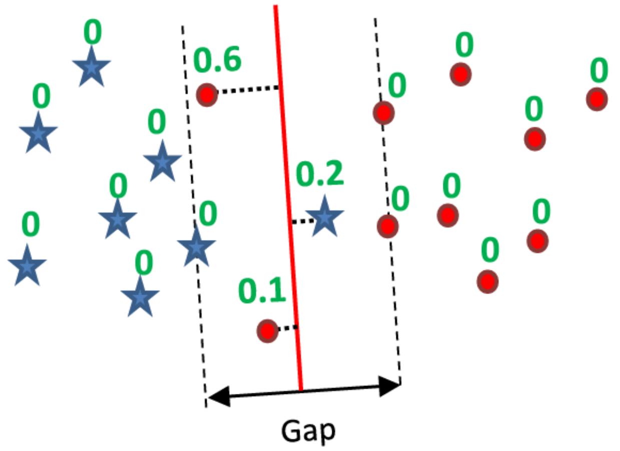
$$L_p = \frac{1}{2}||\vec{\omega}||^2 - \sum_{i=1}^{N} \alpha_i(y_i(\vec{\omega}\vec{x}+b)-1) = \frac{1}{2}||\vec{\omega}||^2 - \sum_{i=1}^{N} \alpha_i(y_i(\vec{\omega}\vec{x}+b) + \sum_{i=1}^{N} \alpha_i$$

$$\begin{cases} \frac{\delta L_p}{\delta b} = 0 \\ \frac{\delta L_p}{\delta \vec{\omega}} = 0 \end{cases} \Leftrightarrow \begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ \vec{\omega} = \sum_{i=1}^{N} \alpha_i y_i x_i \end{cases} \tag{7}$$

Այստեղից կգտնենք $\vec{\omega}$ վեկտորը:

Դեպք 2. Երբ մեր դասերը չենք կարողանում բաժանել գծորեն, այսինքն տվյալներ են լինում որոնք սխալ են դասակարգված, յուրաքանչյուր տվյալի համար նոր փոփոխական ենք ստեղծում, որը ցույց է տալիս թե տվյալը ինչքան հեռու է հիպերպլանից: Այդ հեռավորությունը նշանակենք ϵ_i-ով: $\epsilon_i \geq 0$, այն հավասար է 0-ի, եթե մեր տվյալը ճիշտ է դասակարգված: $0 < \epsilon_i < 1$, եթե այն ճիշտ է դասակարգվել, բայց ընկած է Gap-ի մեջ:Այսինքն պետք է ունենանք, որ՝

$$\begin{cases} \vec{\omega}\vec{x_i}+b \leq -1+\epsilon_i, \text{ եթե } y_i = -1 \\ \vec{\omega}\vec{x_i}+b \geq 1-\epsilon_i, \text{ եթե } y_i = +1 \end{cases} \tag{8}$$



Կամ նույնն է,որ գրենք՝

$$y_i(\vec{\omega}\vec{x}_i + b) \geq 1 - \epsilon_i \forall i = 1, \ldots, N$$

Դասակարգումը կկապարվի հետևյալ ֆունկցիայով՝

$$f(\vec{x}) = sign(\vec{\omega}\vec{x}_i + b)$$

Այս դեպքում մենք ցանկանում ենք մինիմալացնել

$$\frac{1}{2}||\vec{\omega}||^2 + C\sum_{i=1}^{N}\epsilon_i$$

C-ն ցույց է փալիս թե ինչ կարևորություն ենք փալիս շեղումներին: Այսպեղ մենք նաև ցանկանում ենք, որ սխալները քիչ լինեն: Այսինքն մեր առջև նոր խնդիր է դրվում՝ մինիմալացնել $\frac{1}{2}||\vec{\omega}||^2 + C\sum_{i=1}^{N}\epsilon_i$ այն պայմանով, որ $\epsilon_i \geq 0$ և $y_i(\vec{\omega}\vec{x}_i + b) \geq 1 - \epsilon_i$: Լագրանժի ֆունկցիան կլինի՝

$$L_p = \frac{1}{2}||\vec{\omega}||^2 + C\sum_{i=1}^{N}\epsilon_i - \sum_{i=1}^{N}\alpha_i(y_i(\vec{\omega}\vec{x} + b) - 1) - \sum_{i=1}^{N}\mu_i\epsilon_i$$

որպեղ μ_i-երը Լագրանժի բազմապապկիչներն են, որոնք կիրառվում են ϵ_i-երի ոչ բացասական լինելու համար: Գփնելով այս ֆունկցիայի մինիմումը, կսփանանք լուծումը:

Մոդելի ստուգման եղանակներ

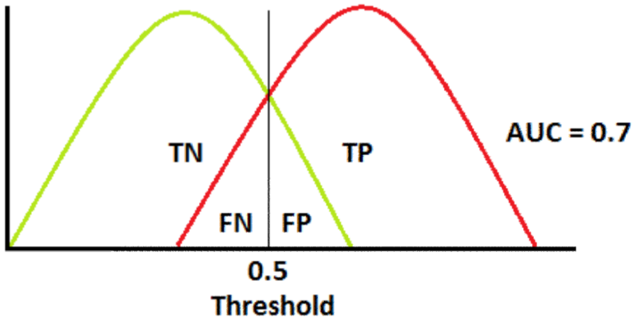
Որպեսզի ստուգենք մոդելի ճշգրտությունը մենք մեր տվյալները բաժանում ենք 2 մասի՝ $train$-ի, որի վրա կիրառում ենք մեր ալգորիթմները և $test$-ի, որի վրա ստուգում ենք մեր մոդելները: Մոդելի ճշգրտությունը չափելու համար մի շարք ցուցանիշներ կան:

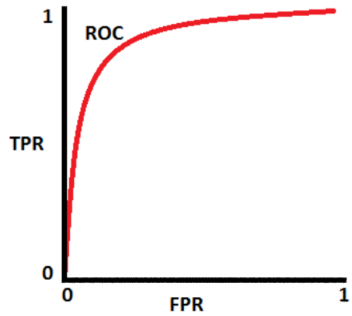
$Confusion\ matrix$

TP- Այն տվյալների քանակն է, որոնք "այո" են և մենք "այո" էլ կանխատեսել ենք ($True\ Positive$)
TN-Այն տվյալների քանակն է, որոնք "ոչ" են և մենք "ոչ" էլ կանխատեսել ենք ($True\ Negative$)
FP- Այն տվյալների քանակն է, որոնք "ոչ" են, սակայն մենք "այո" ենք կանխատեսել ($False\ Positive$)
FN- Այն տվյալների քանակն է, որոնք "այո" են, սակայն մենք "ոչ" ենք կանխատեսել ($False\ Negative$)
$Accurancy\ score = \frac{TP+TN}{All}$
$True\ positive\ rate(TPR) = \frac{TP}{TP+FN}$
$False\ positive\ rate(FPR) = \frac{FP}{FP+TN}$
$Sensitivity = \frac{TP}{TP+FN}$
$Specificity = \frac{TN}{TN+FP} = 1 - FPR$

$ROC\ Curve\ (Receiver\ Operating\ Characteristics)\ AUC\ (Area\ Under\ The\ Curve)$

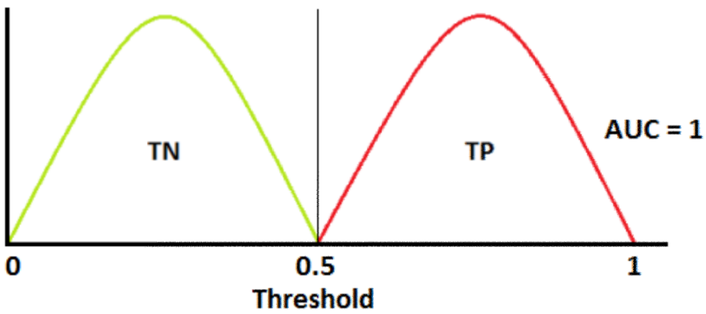
ROC կորը հավանականային կոր է, իսկ AUC-ը ցույց է տալիս բաժանելիության աստիճանը, ինչքան մեծ է AUC-ը այնքան մեր մոդելը լավ է ճանաչում "այո"-ն որպես "այո",իսկ "ոչ"-ը՝ "ոչ": ROC կորը գծում ենք FPR-ն x-երի, իսկ TPR-ն՝ y-ների առանցքի վրա: Տեսնենք ինչպես են գտնում ROC կորը: Ցույց տանք գրաֆիկորեն, գծենք TP-ի և TN-ի հավանականային կորերը:

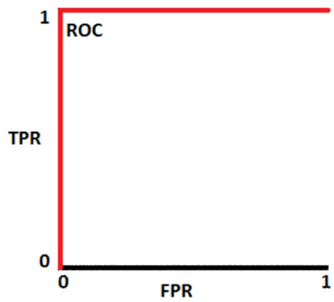




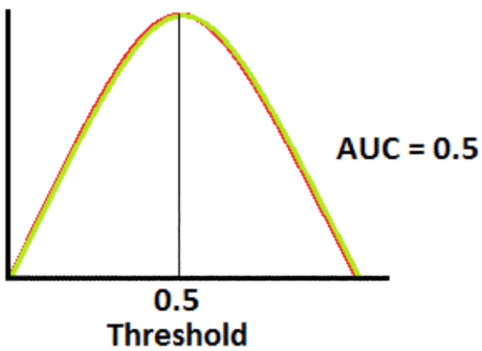
$Threshold$-ը տեղաշարժելով կստանանք TPR-ի և FPR կոմբինացիաներ, որոնք գրաֆիկորեն պատկերելով կստանանք ROC կորը: AUC-ը այդ կորից ներքև ընկած պատկերի մակերեսն է: Եթե օրինակ $AUC = 0.7$ նշանակում է 70 տոկոս հավանականություն կա, որ մոդելը կտարբերի ”այո”-ն ”ոչ”-ից:

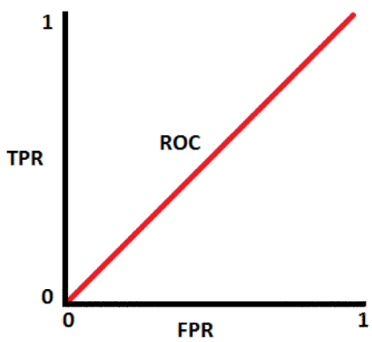
Լավագույն դեպքում՝

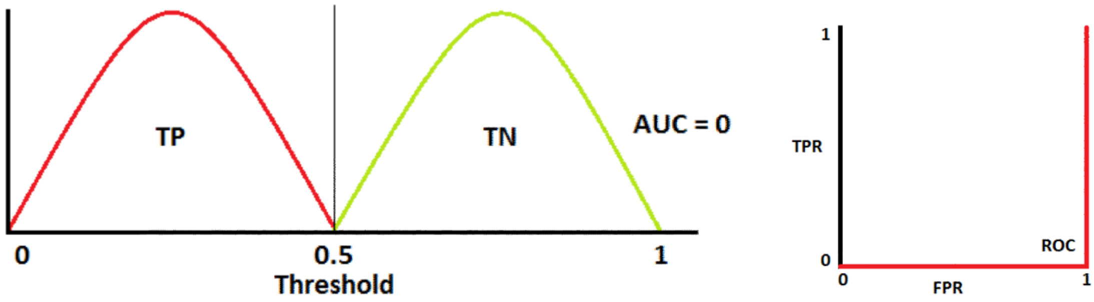




Վատագույն դեպքում՝







Այս դեպքում մոդելը ”այո”-ն ”ոչ” է կանխատեսում, իսկ ”ոչ”-ը՝ ”այո”

KFold Cross Validation

Հնարավոր է, որ բաժանելուց մեր փվյալները վատ բաժանենք և թեստային փվյալները վատ կանխատեսվեն, բայց իրականում մեր մոդելը լավը լինի: Այդ պատճառով հաշվում ենք $K - Fold\ Cross - Validation\ score$-ը: Սկզբից մենք մեր փվյալները բաժանում ենք K մասի և վերցնում ենք այդ K մասերից $K - 1$-ը որպես *train* և հաշվում մոդելի որակը սպուգող որևէ գնահատական: Այնուհետև որպես *test* վերցնում ենք մեկ այլ մաս և դրա համար հաշվում գնահատականը: Այդպես շարունակ բոլոր մասերի համար և վերջում միջինացնում ենք այդ գնահատականները: Եվ մոդելը սպուգելու համար օգտագործում ենք այդ միջինացված արժեքը:

Եզրակացություն

Կիրառելով տրված 6 ալգորիթմները ստացել ենք հետևյալ արդյուքները: (Այստեղ կտեղադրենք արդյունքների աղյուսակներ,գրաֆիկներ)

Արդյունքում բավականին լավ ցուցանիշներ են ստացվել: Հետագայում կարելի է կիրառել ավելի բարդ ալգորիթմներ և համեմատել մեր ստացված արդյունքների հետ:

բլաբլաբլա

Հավելված

Այսպեղ կպեղադրենք $Github$-ի հղումը