Introduction to k-Nearest Neighbors(k-NN)

k Nearest Neighbor (k-NN) သည် ပေါ်ပြူလာဖြစ်သည့် supervised machine learning algorithm တစ်မျိုး ဖြစ်သည်။ k-NN algorithm သည် instance-based learning algorithm ၊ competitive learning algorithm အမျိုးအစား ဖြစ်သည်။ Train data များကို သိမ်းရုံသာသိမ်းထားပြီး ကြိုတင် ပြင်ဆင်ထားခြင်း မရှိဘဲ test ဒေတာ ထည့်ပေးသည့်အချိန်မှ အလုပ် ထလုပ်သောကြောင့် lazy learning algorithm ဟုလည်း ခေါ်ဆိုလေ့ရှိသည်။

k-NN Algorithm သဘော တရား နှင့် input parameter များက output သို့မဟုတ် ခန့်မှန်းချက် (prediction)ကို မည်ကဲ့သို့ အကျိုးသက်ရောက်မှု ရှိစေသည်ကို လေ့လာကြမည်။

အသုံးပြုပုံ

Classification ပြဿနာ နှင့် regression ပြဿနာ နှစ်မျိုးလုံး အတွက် k-NN ကို အသုံးပြုကြသော်လည်း လက်တွေ့ လုပ်ငန်းခွင်များတွင် classification ပြဿနာများအတွက်သာ အလွန်အသုံးများသည်။ k-NN ကို regression ပြဿနာ အတွက် သုံးသည့်အခါ ပျမ်းမျှတန်ဖိုး(the mean or the median of the k-most similar instances)ကို ယူသည်။

k-NN algorithm မှ ထွက်လာသည့် အဖြေများ မှန်သည် မှားသည်ကို အလွယ်တကူ သိနိုင်သည်။ (ease to interpret output)။ တွက်ချက်သည့်အချိန်(calculation time) တိုတောင်းသည်။ တစ်ခါတစ်ရံ တိကျမှု(predictive power)သည် တခြားသော algorithm များနှင့် နှိုင်းယှဉ် အသင့်အတင့်သာ ကောင်းနိုင်သည်။

k-NN algorithm သဘောတရား

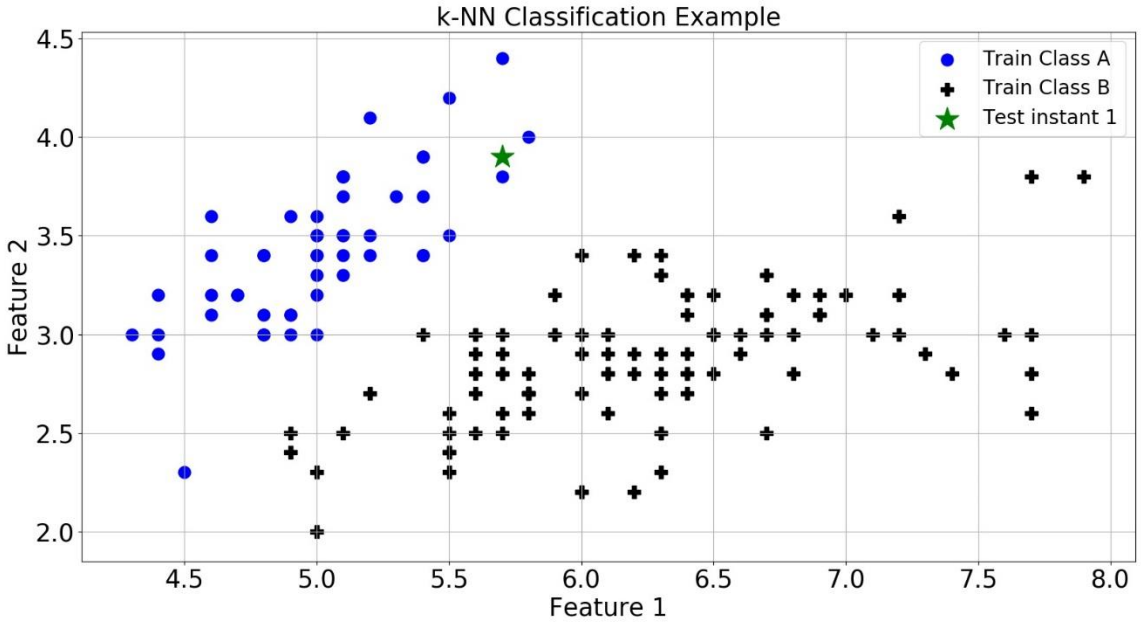
k-NN algorithm ကို ကောင်းစွာ နားလည်နိုင်ရန် အောက်ပါအတိုင်း အရိုးရှင်းဆုံး ဥပမာဖြင့် ဖော်ပြနိုင်သည်။ သင့်အား "ဘယ်မှာနေသလဲ" ဟုမေးလျှင် မိမိနေသည့် အိမ်နေရာနှင့် အနီးစပ်ဆုံးမြို့ကို ရှာဖွေ၍ မော်လမြိုင်မှ နေသည်၊ ဖားအံမှာ နေသည် စသည်ဖြင့် ပြောဆိုလေ့ ရှိသည်။ ထို့အတူ အနီးစပ်ဆုံးရှိနေသည့် အစုဝင်များကို ရှာဖွေ၍ ထိုအစု(class)ထဲတွင် ပါဝင်သည်ဟု သတ်မှတ် ဆုံးဖြတ်သည်။

နောက်ဥပမာတစ်ခုမှ လူတစ်ယောက်သည် ကိုရီးယား လူမျိုးနှင့် ခပ်ဆင်ဆင်တူလျှင် ကိုရီယား ဖြစ်နိုင်သည်ဟု ခန့်မှန်းကြသည်။ ဂျပန်လူမျိုးနှင့် ခပ်ဆင်ဆင်တူလျှင် ဂျပန်ဖြစ်နိုင်သည်ဟု ခန့်မှန်းကြသည်။ ထို့အတူ k-NN algorithm တွင် ဒေတာများ တူညီမှု(similarity)ကို အဓိကထား၍ မည်သည် အစုအဖွဲ့ (class)တွင် ပါဝင်သည်ဟု ခန့်မှန်းခြင်း ဖြစ်သည်။

တူညီမှု(similarity)ကို ဂရပ်ဖော်တွင် ဖော်ပြ ပြောဆိုလျှင်၊ သင်္ချာနည်းဖြင့် ဖော်ပြလိုလျှင် အကွာအဝေး (distance)ကို အသုံးပြု ရသည်။ အနီးစပ်ဆုံး(distance အနည်းဆုံး) ဖြစ်လျှင် အတူညီဆုံး ဖြစ်သည်။ ဤအချက်သည် k-NN algorithm ၏ တစ်ခုတည်းသော အဓိက ယူဆချက်(assumption) ဖြစ်သည်။

ထိုအတူ k-NN algorithm တွင်လည်း သိလိုသည် input ဒေတာ တစ်ခု (instance တစ်ခု) ပါဝင်နေသည့် အစု(class)ကို သိရန် ခွဲခြားရန်အတွက် ထို input ဒေတာ တစ်ခုနှင့် အနီးစပ်ဆုံး အစုကို ရှာဖွေပြီး မည်သည့်အစု(class) တွင် ပါဝင်သည့်ဟု k-NN algorithm က ခန့်မှန်း ပေးသည်။

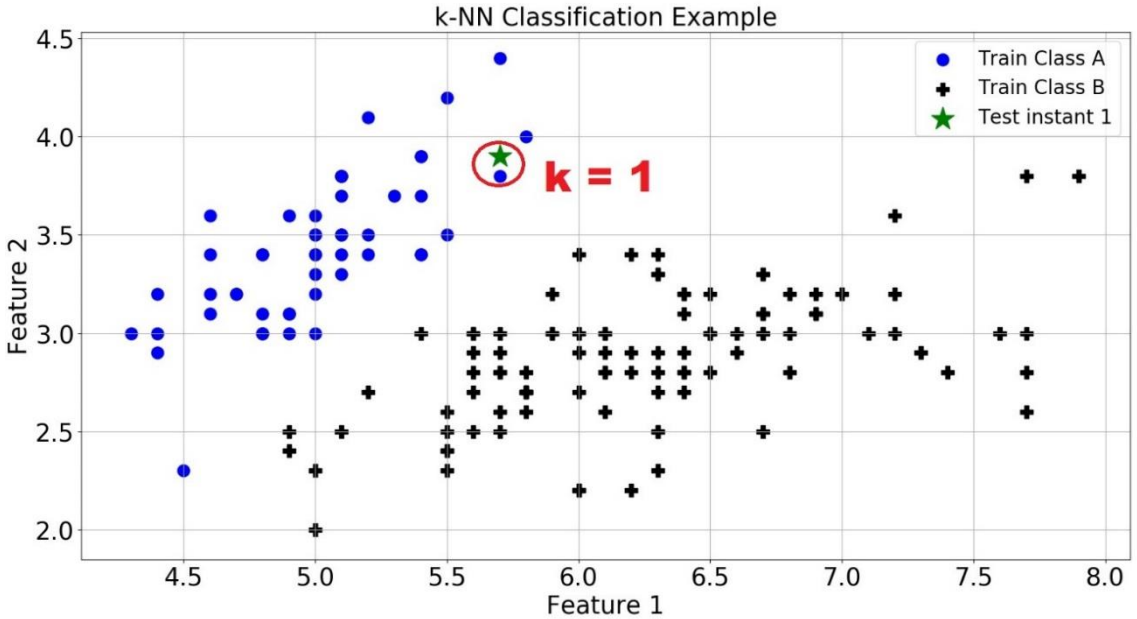
k-NN algorithm ကို ပိုနားလည် သဘောပေါက်စေရန် ပုံဖြင့် သရုပ်ဖော် ရှင်းပြထားသည်။



ပုံ - ၁ k-NN classification algorithm ဥပမာ

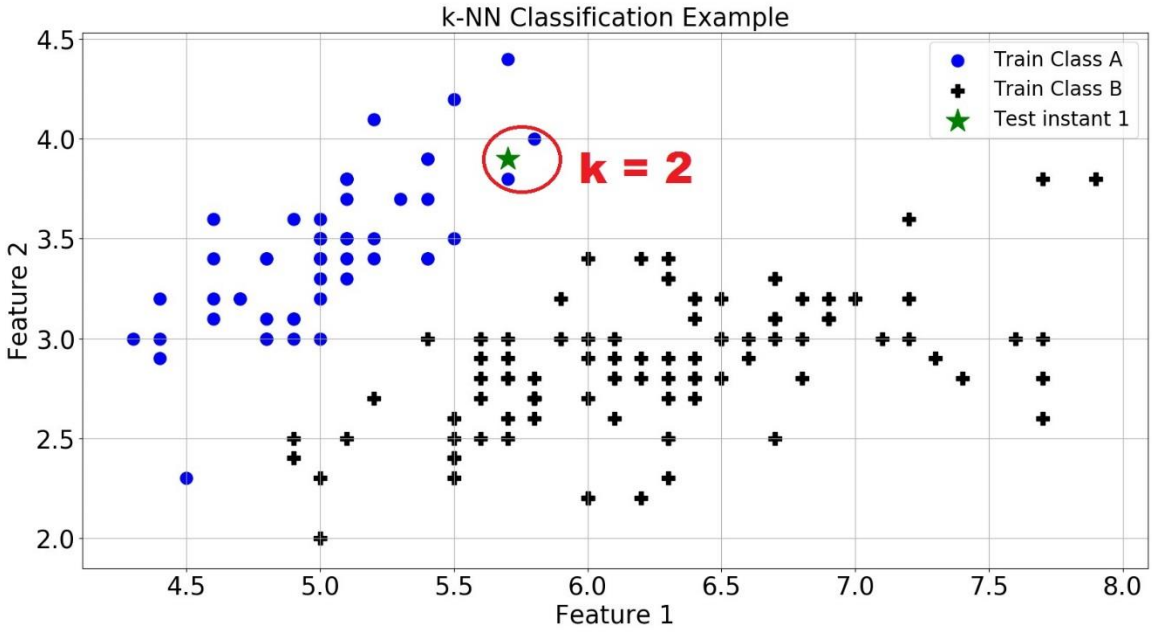
အပြာရောင် အစက်များသည် Train class A ဖြစ်ပြီး အပေါင်းလက္ခဏာ သင်္ကေတသည် Train class B ဖြစ်သည်။ အစိမ်းရောင် ကြယ်ပွင့်သည် Test instance 1 ဖြစ်သည်။ Test instant 1 သည် Train class A တွင် ပါဝင်နေကြောင်း မြင်ရုံဖြင့် သိနိုင်သည်။

သို့သော် k-NN algorithm သဘောအရ အရေအတွက်(k) မည်မျှဖြင့် အနီးစပ်ဆုံး တူညီရမည်ဆိုသည့် သတ်မှတ်ချက်ကြောင့် k အရေအတွက်ကို ဆုံးဖြတ်ပေးရသည်။



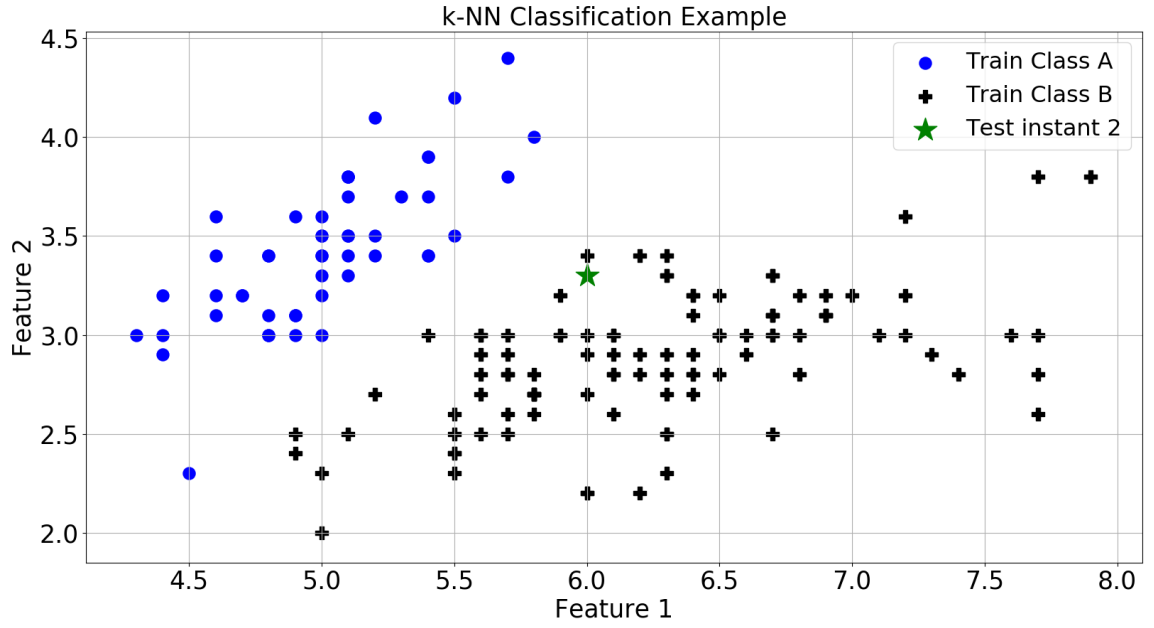
ပုံ - ၂ k-NN classification algorithm ဥပမာ (k = 1)

ဥပမာ Test instant 1 နှင့် အနီးစပ်ဆုံးတစ်ခုကို ရှာမည်ဆိုလျှင် အပေါ်ပုံတွင် ပြထားသည့် အတိုင်းဖြစ်သည်။ k အရေအတွက် သည် k =1 ဖြစ်သည်။



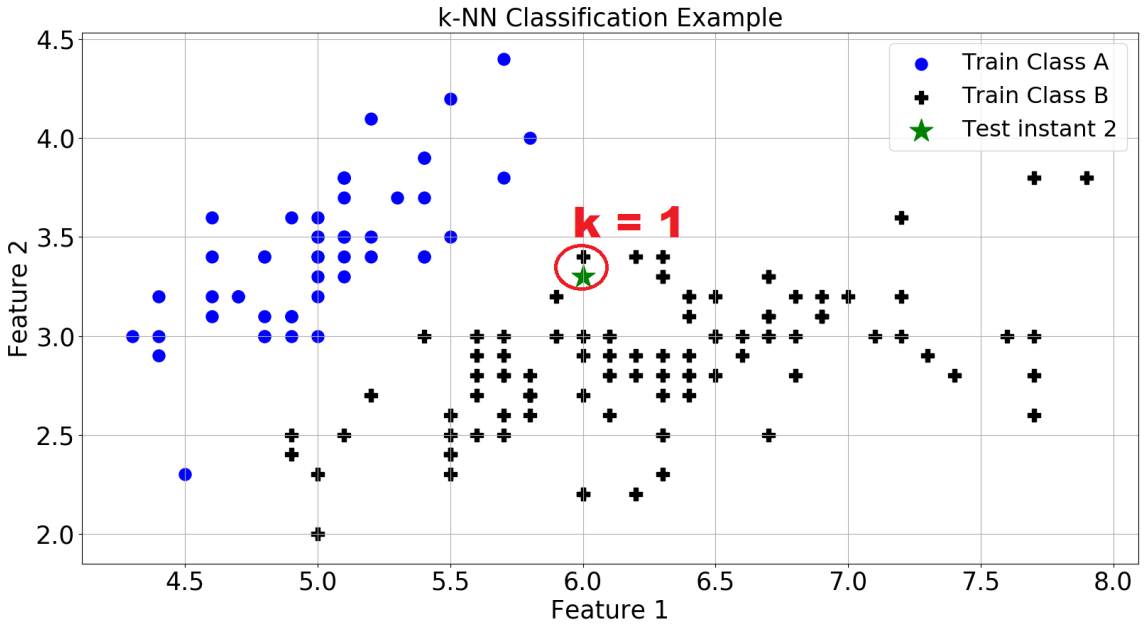
ပုံ - ၃ k-NN classification algorithm ဥပမာ (k = 2)

ဥပမာ Test instant 1 နှင့် အနီးစပ်ဆုံး နှစ်ခုကို ရှာမည်ဆိုလျှင် အပေါ်ပုံတွင် ပြထားသည့် အတိုင်းဖြစ်သည်။ k အရေအတွက်သည် k =2 ဖြစ်သည်။

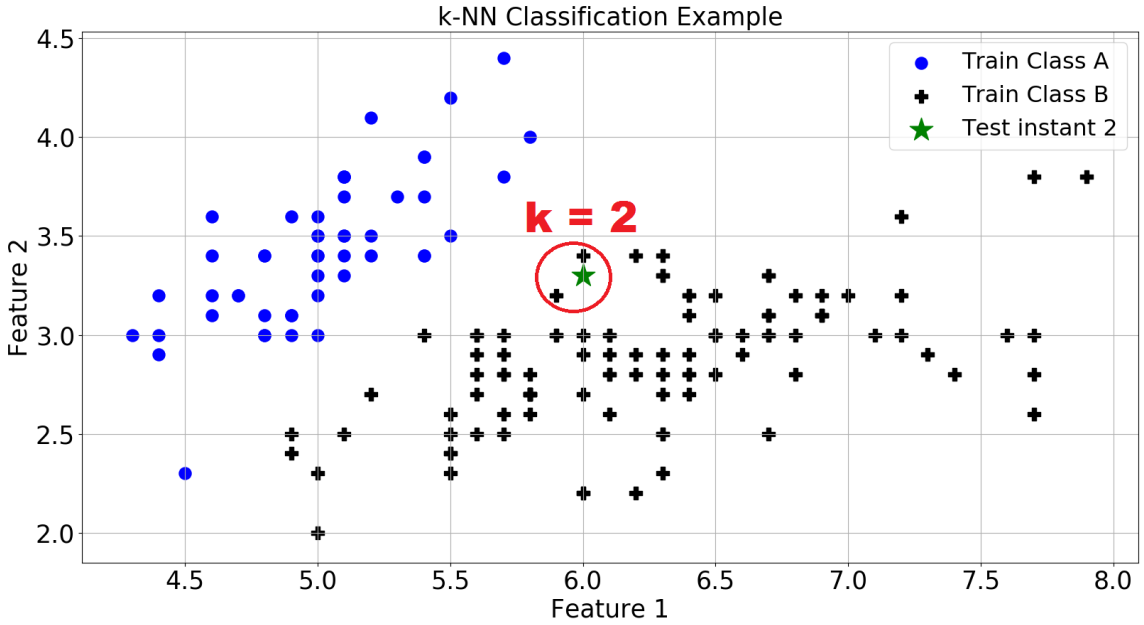


ပုံ - ၄ k-NN classification algorithm ဥပမာ

ဥပမာ Test instant 1 နှင့် အနီးစပ်ဆုံးတစ်ခုကို ရှာမည်ဆိုလျှင် အပေါ်ပုံတွင် ပြထားသည့် အတိုင်းဖြစ်သည်။ k အရေအတွက် သည် k =1 ဖြစ်သည်။



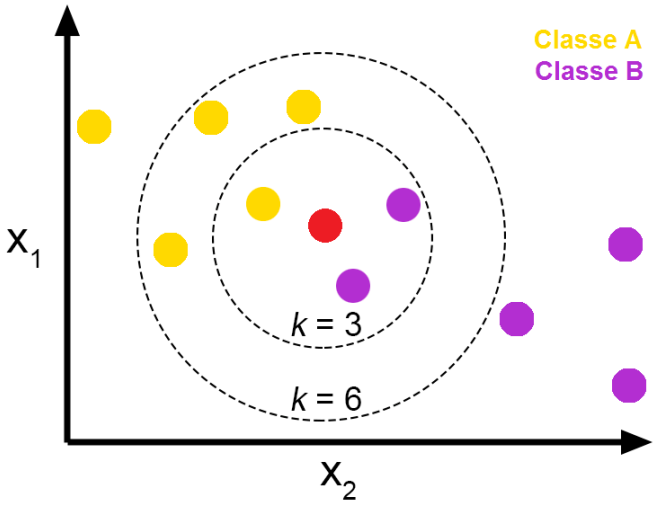
ပုံ - ၅ k-NN classification algorithm ဥပမာ (k = 1)



ပုံ - ၆ k-NN classification algorithm ဥပမာ (k = 2)

K အရေအတွက်

K အရေအတွက် အနည်းအများ သည် algorithm က ခန့်မှန်းပေးသည့် ရလဒ်ကိုမည်ကဲ့သို့ ပြောင်းလဲနိုင်သည်ကို နားလည်ထားရန်လိုသည်။

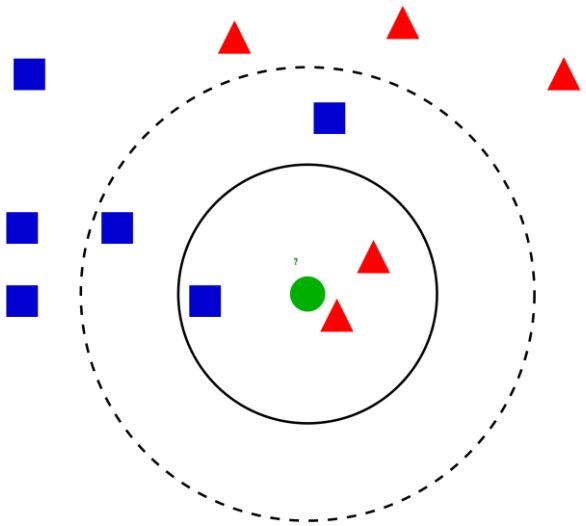


ပုံ - ၇ k-NN classification algorithm ဥပမာ (k = 3 and k = 6)

အထက်က ပုံတွင် k = 3 (အနီးစပ်ဆုံး ၃ ခု)ဖြင့် ဆုံးဖြတ် ခန့်မှန်းမည်ဟု သတ်မှတ်လျှင် အနီအဝိုင်းလေးသည် Class B ထဲတွင် ပါဝင်ပြီး k = 6 (အနီးစပ်ဆုံး 6 ခု)ဖြင့် ဆုံးဖြတ် ခန့်မှန်းမည်ဟု သတ်မှတ်လျှင် အနီဝိုင်းလေးသည် Class A ထဲတွင် ပါဝင်သည်။

ထိုကြောင့် k အရေအတွက်သည် တိကျသည့် ခန့်မှန်းချက်(prediction) ရရှိရန် အဓိကကြသည့် အချက် ဖြစ်သည်။

အစိမ်းရောင် ကြယ်ပွင့်သည် မည်သည့် အစုတွင် ပါဝင်သည်ကို သိလိုပါက အနီးစပ်ဆုံးရှိနေသည့် အရာကို ရှာဖွေနိုင်းယှဉ်ရသည်။ ထိုအပြင် အနီးစပ်ဆုံး ရှိနေသည် အရာများအနက် ပိုများသည့် အစုကို ယူရသည်။

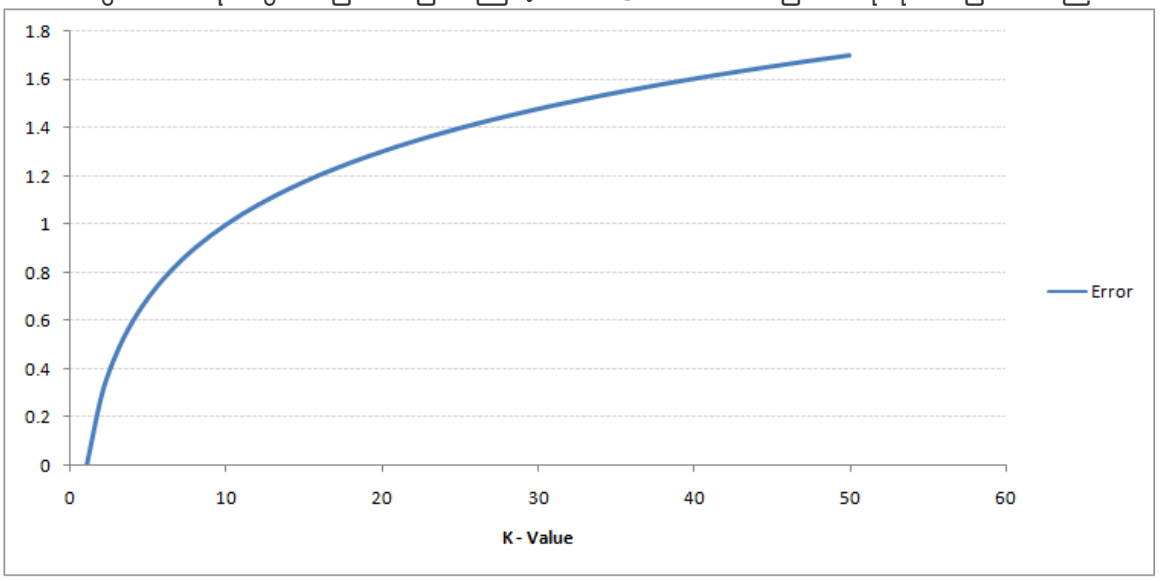


ပုံ - ၈ k-NN classification algorithm ဥပမာ (k = 3 and k =5)

အထက်က ပုံတွင် k=3 နှင့် k=5 တို့ ထုတ်ပေးသည့် ခန့်မှန်းချက် နှစ်ခုတို့မှာ တူညီကြလိမ့်မည် မဟုတ်ပေ။

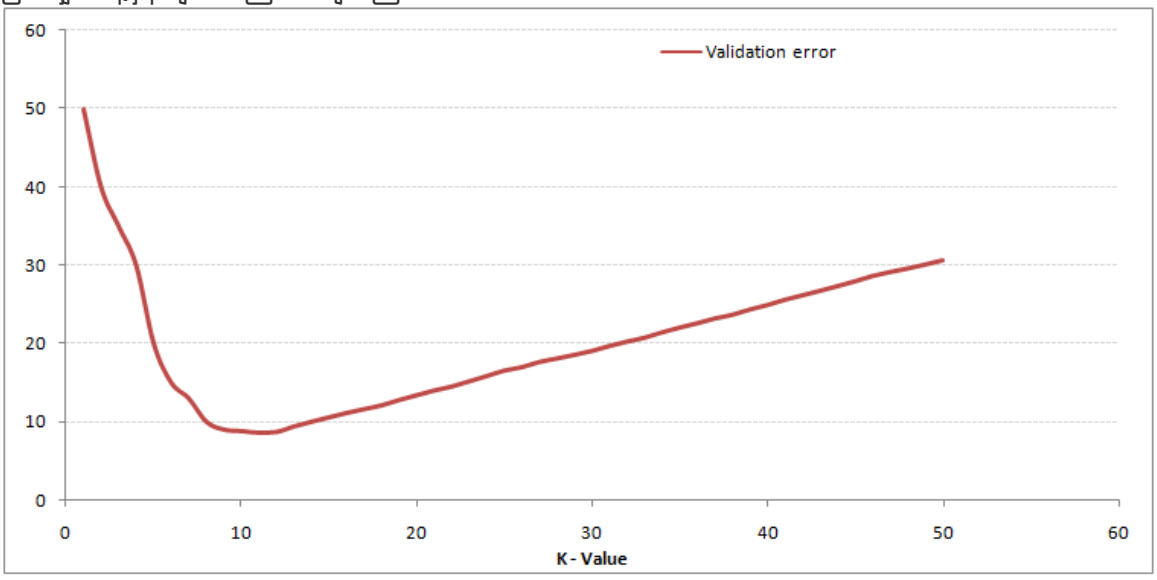
k အရေအတွက် များခြင်း နည်းခြင်းကြောင့် ဖြစ်ပေါ်လာမည့် training error rate နှင့် validation error rate တို့ကို စစ်ဆေးရန် လိုအပ်သည်။

အောက်တွင် k အရေအတွက် ပြောင်းလဲခြင်းကြောင့် training error rate ပြောင်းလဲပုံကို ဖော်ပြထားသည်။



ပုံ - ၉ training error rate with varying value of K

k အရေအတွက် (၁) ဖြစ်လျှင် (K=1) error rate သည် သုည ဖြစ်သည်။ အဘယ်ကြောင့်ဆိုသော် data point နှင့် အနီးစပ်ဆုံး လုံးဝ တူညီသည် point သည် ထို point ကိုယ်တိုင်ပင်ဖြစ်သည်။ ထို့ကြောင့် K=1 ဖြစ်လျှင် ခန့်မှန်းချက်သည် တိကျသည်။(prediction is always accurate)



ပုံ - ၁၀ training error rate with varying value of K

အပေါ်ပုံ (ပုံ-၁၀)တွင် k အရေအတွက်ကို လိုက်၍ validation error ပြောင်းလဲပုံကို ဖော်ပြထားသည်။ k အရေအတွက်(၁၀ခု)တွင် validation error အနည်းဆုံးဖြစ်ပြီးနောက် k အရေအတွက် များလာလေ validation error ပိုများလေ ဖြစ်သည်။

k-NN algorithm သည် train time အချိန်တွင် ဒေတာများကို သိမ်းရုံသာ သိမ်းဆည်းထားသည်။ test time အချိန်တွင် သာ computation လုပ်သည်။ algorithm ၏ training phase တွင် training samples များ၏ ဒေတာများကို feature vectors နှင့် class labels အစီအစဉ်တကျ သိမ်းထားရုံသာ လုပ်သည်။

testing phase တွင် test data နှင့် training data တစ်ခုချင်းစီ(each row) တို့၏ အကွာအဝေး(distance)ကို တွက်သည်။ များသော အားဖြင့် Euclidean distance နည်းကို အသုံးပြုသည်။ တခြားသော အကွာအဝေး(distance) ရှာဖွေသည့်နည်းများကိုလည်း အသုံးပြုနိုင်သည်။ အကွာအဝေး အများအနည်း အပေါ်ကို အခြေခံ၍ အနီးစပ်ဆုံးမှ အဝေးဆုံးသို့ အစဉ်အတိုင်း training data များကို စဉ်သည်။ ထိပ်ဆုံးမှ အနီးဆုံး k အရေအတွက် row များအတိုင်း ခန့်မှန်းချက် ထုတ်ပေးသည်။ နီးစပ်မှု အများဆုံးရှိသည့် အစုကို ရွေးချယ်သည်။

အကွာအဝေး(distance between two points) ရှာနည်းများမှာ

ပွိုင့် နှစ်ပွိုင့် အကြားအကွာဝေး(distance between two points)ကို တိုင်းယူနိုင်သည့် နည်းများအားလုံးကို သုံးနိုင်သည်။ အကွာအဝေးတိုင်းသည့်နည်းများမှာ Manhattan နည်း ၊ Minkowski နည်း၊ Tanimoto နည်း ၊ Jaccard နည်း ၊ Mahalanobis နည်းများ ဖြစ်ကြသည်။

(က) Manhattan distance
(ခ) Minkowski distance
(ဂ) Tanimoto distance
(ဃ) Jaccard distance
(င) Mahalanobis distance

မှတ်ရန်အချက်များ

k-NN algorithm ကို အသုံးပြုရန် data များ အားလုံး၏ စကေး(scale) တူညီရမည်။ နီးစပ်မှုရှိသည်။ closed distance စသည်တို့ကိုသာ စဉ်းစားသည်။ functional အတွက် မည်သည့် assumption မှ လုပ်ရန်မလိုပါ။

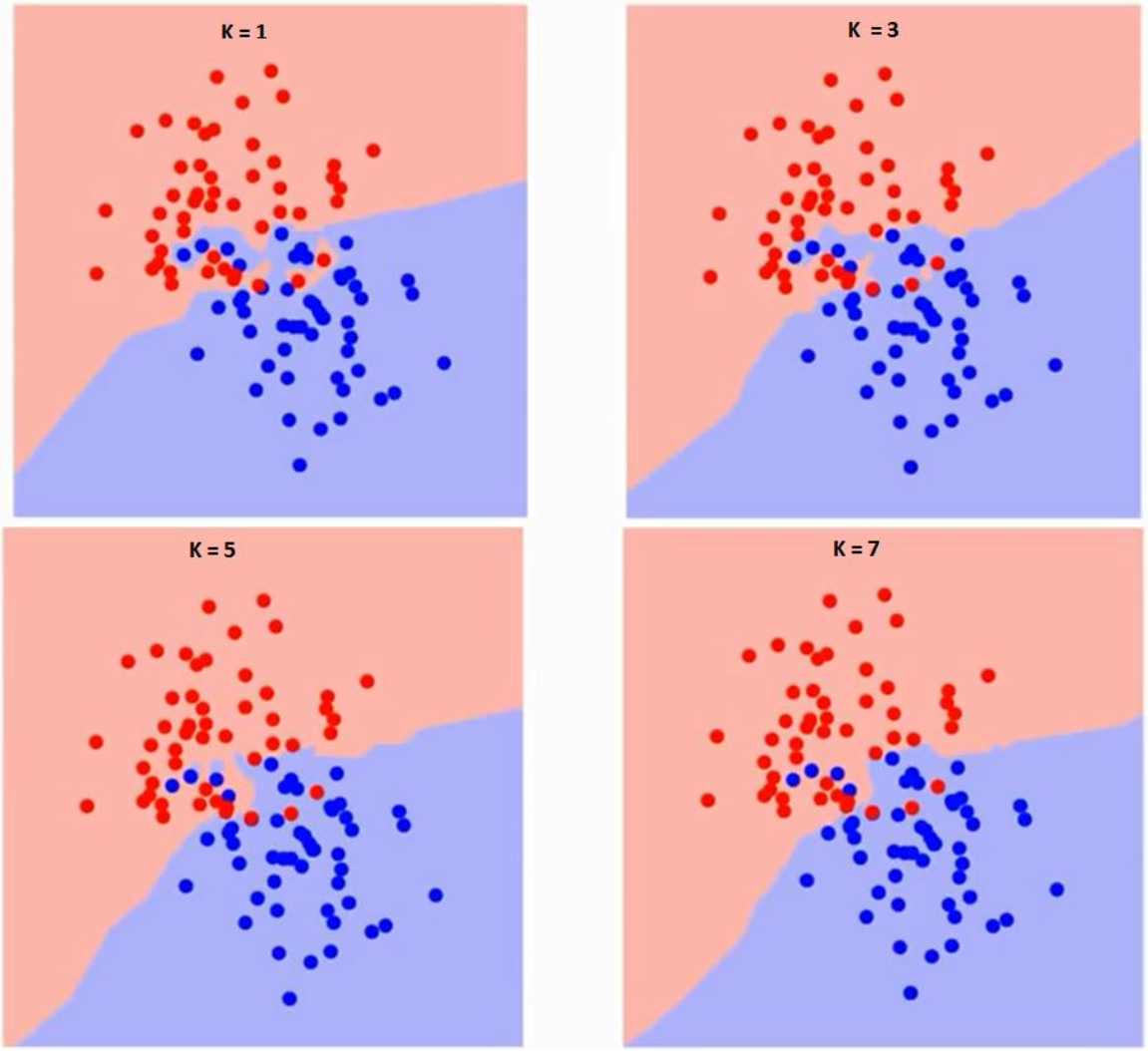
Categorical variables နှင့် continuous variables များတွင် ဖြစ်ပေါ်သည့် missing value များ ပြန်ဖြည့်ရန်အတွက် K-NN algorithm ကို အသုံးပြုနိုင်သည်။

K အရေအတွက် များလာလေ bias များလာလေဖြစ်သည်။ (When you increase the k the bias will be increases)။ K အရေအတွက် နည်းလေ variance များလာလေဖြစ်သည်။ (When you decrease the k the variance will increases)

ဒေတာတွင် noise များ ရှိနေလျှင် ပိုများသည့် k အရေအတွက်(increase the value of k)ကို အသုံးပြုသင့်သည်။(noise in data increase the value of k)

k-NN အတွက် explicit training step မလိုအပ်ပါ။

K အရေအတွက်များ ခြင်းကြောင့် decision boundary ပို ပြေပြစ်(smother)လိမ့်မည်။



အစုနှစ်ခုကြား ပိုင်းခြားထားသည့် လိုင်းကို decision boundary ဟုခေါ်သည်။ k အရေအတွက်များလာလေ(increasing value of K) decision boundary လိုင်း ပို၍ ပြေပြစ်လာလေ(becomes smoother) ဖြစ်သည်။ (The boundary becomes smoother with increasing value of K)။

Cross validation အကူအညီဖြင့် သင့်လျော်ဆုံး K အရေအတွက်(optimal value of k) ကို ရွေးချယ်နိုင်သည်။

Euclidean distance ကို သုံး၍ အကွာအဝေးဖြင့် ဆုံးဖြတ်သောကြောင့် feature များအားလုံးသည် အညီအမျှ အရေးပါသည်။ (Euclidean distance treats each feature as equally important) ။ တစ်နည်းအားဖြင့် feature များအားလုံးသည် အညီအမျှ အရေးပါမှသာ Euclidean distance ကို အသုံးပြုနိုင်သည်။

Cross validation ကို အကူအညီဖြင့် အကောင်းဆုံး k အရေအတွက်(optimal value of k)ကို ရှာဖွေနိုင်သည်။

မည်သည့် အခြေအနေတွင် K အရေအတွက်(value) မည်မျှကို သုံးသင့်သနည်း။

အောက်ပါ အခြေအနေများတွင် (၁) ထက် ပိုများသည့် K အရေအတွက် သုံးသင့်ပါသည်။ (k valve more than one)

(၁) Attribute တွင် noise များ ရှိနေလျှင် (noise in Attribute)

(၂) Class တွင် noise များ ရှိနေလျှင် (noise in class label)

(၃) Class များ ထပ်နေသည့်အခါ (class overlapping)

တခြားသော အချက်များ

Attribute များအားလုံးသည် အညီအမျှ အရေးပါကြသည်။ (attributes are equally important)

Attribute များ၏ scale သည် တစ်ခုနှင့် တစ်ခု အလွန် မတူညီသည့်အခါ (different attribute scale)

မည်သည့် အခြေအနေတွင် အကွာအဝေး(distance) ရှာသည့် simple eucali distance equation ကို အသုံးပြု၍ ရနိုင်သနည်း။

1. Equal weight to all attribute , only if scale attribute are similar.
2. Scale attribute to equal range
3. Classes are spherical in shape.

စတဲ့အချက်တွေနဲ့ ပြည်စုံပါမှ simple Euclidean distance equation ကို အသုံးပြုနိုင်ပါသည်။

အကယ်၍ classes များသည် spherical in shape ဖြစ်မနေသည့်အခါမျိုး၊ အကယ်၍ attribute တစ်ခု တခြားသော attribute တစ်ခုထက် ပို၍ အရေးပါ(အကျိုးသက်ရောက်မှု ပိုများ)နေသည့်အခါ နှင့် အကယ်၍ attribute မှ noise တွေ ရှိနေသည့်အခါ များတွင် အကွာအဝေး(distance) ရှာသည့် simple eucali distance equation ကို အသုံးပြု၍ မရနိုင်ပါ။ ထိုအခြေအနေများတွင် weight Euclidean distance ကို အသုံးပြုသင့်သည်။

လိုအပ်လျှင် attribute ၏ scale များကို ညှိသင့်သည်။ (normalize the attribute)

k အရေအတွက် မည်မျှ ဖြစ်သင့်သည်ကို ဆုံးဖြတ်ရန်အချက်များ

k အရေအတွက် နည်းလျှင် စဉ်းစားနေသည့် ပြဿနာ၏ အသေးစိတ်များကို သိနိုင်သည်။

(small k → capture very fine structure of problem)

Train data set တွင် ဒေတာ အနည်းငယ်သာ ရှိသည့်အခါ နည်းသည့် k အရေအတွက်(small k value) ကို အသုံးပြုနိုင်သည်။

များသည့် k အရေအတွက်(larger k value)

အထူးသဖြင့် class noise ရှိနေသည့်အခါ များသည့် k အရေအတွက်(large k)ကို အသုံးပြုသင့်သည်။(less sensitive to noise)

Training data အလွန်များပြားနေသည့်အခါ(large training set ဖြစ်သည့်အခါ) larger k value ကို အသုံးပြုနိုင်သည်။

ယေဘုယျအားဖြင့် k အရေအတွက်သည် (၅) ထက်နည်းလျှင် small k valueဟု သတ်မှတ်၍ (၅) ထက် ပိုများလျှင် (larger k value) ဟုသတ်မှတ်သည်။

Feature reduction

အချို့သော feature သည် တခြားသော feature ထက် ပို၍ အကျိုးသက်ရောက်မှုများသည့်အခါ(more important) နှင့် တချို့သော feature သည် အကျိုးသက်ရောက်မှု လုံးဝ မရှိသည့်အခါ (no important at all) မျိုးတို့တွင် Feature reduction လုပ်ရန်လိုသည်။

-End-