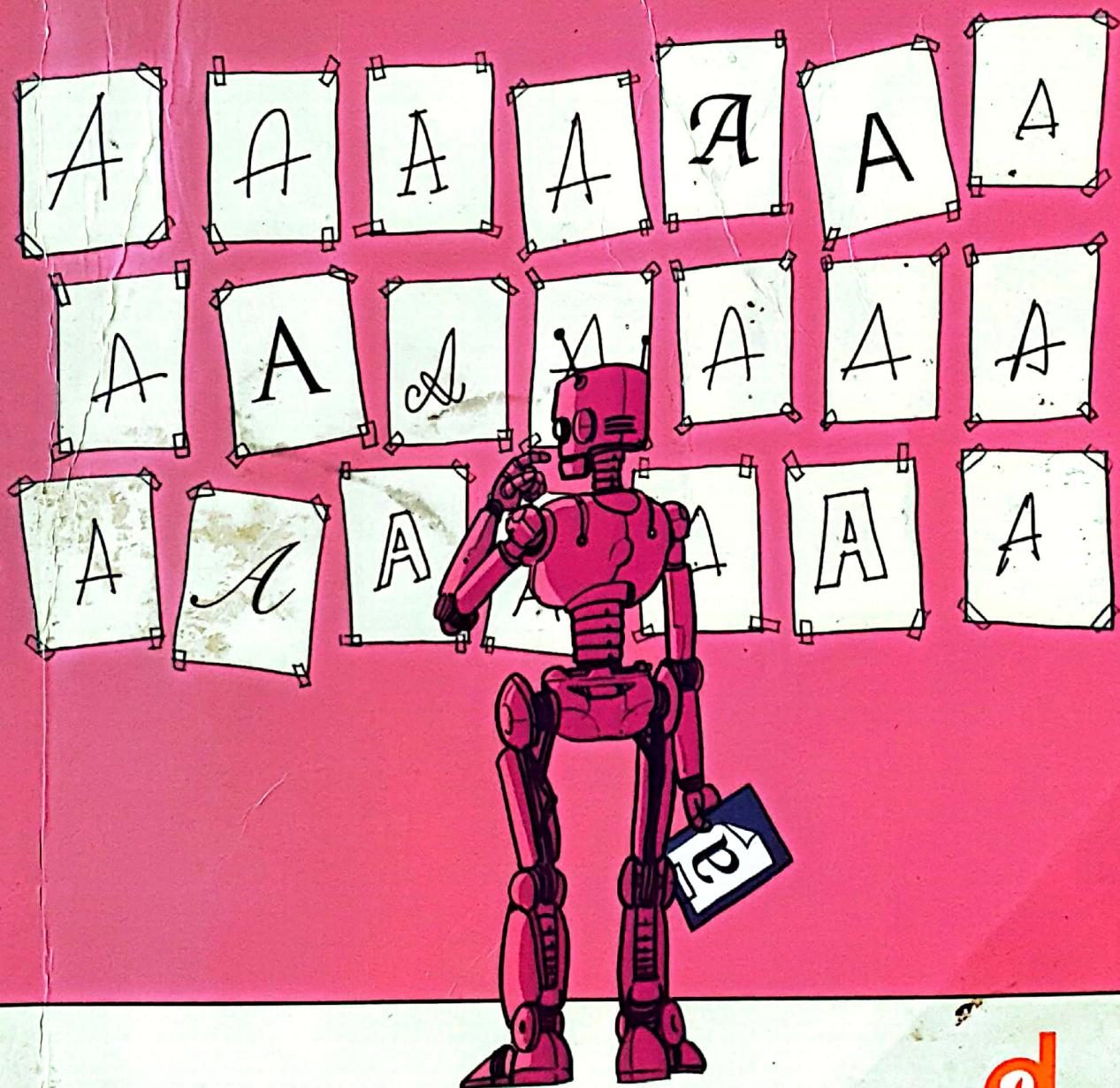


মেশিন লার্নিং অ্যালগরিদম

নাফিস নিহাল



দ্বিমিক প্রকাশনী

সূচি

মুখ্যবন্ধন	১১
ভূমিকা	১৩
লেখক পরিচিতি	১৭
অধ্যায় ১ : কৃত্রিম বুদ্ধিমত্তা এবং মেশিনের যত কথা	১৯
অধ্যায় ২ : বিভিন্ন ধরনের লার্নিং ও অন্যান্য	২৫
পরিচেদ ২.১ : সুপারভাইজড ও আনসুপারভাইজড লার্নিং (Supervised and Unsupervised Learning)	২৬
পরিচেদ ২.২ : ফিচার (Feature)	২৮
পরিচেদ ২.৩ : ট্রেনিং, টেস্ট ও ভ্যালিডেশন ডেটা (Training, Test and Validation Data)	৩১
পরিচেদ ২.৪ : ক্রস ভ্যালিডেশন (Cross Validation)	৩৪
অধ্যায় ৩ : লিনিয়ার রিপ্রেশন (Linear Regression)	৩৭
পরিচেদ ৩.১ : লিনিয়ার রিপ্রেশন কী	৩৭
পরিচেদ ৩.২ : নিউমেরিকাল অ্যানালাইসিস (Numerical Analysis) ব্যবহার করে লিনিয়ার রিপ্রেশন	৪০
পরিচেদ ৩.৩ : গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম (Gradient Descent Algorithm)	৪৭
পরিচেদ ৩.৪ : গ্রেডিয়েন্ট ডিসেন্ট (Gradient Descent) অ্যালগরিদম ব্যবহার করে লিনিয়ার রিপ্রেশন	৫৩
পরিচেদ ৩.৫ : গ্রেডিয়েন্ট ডিসেন্ট (Gradient Descent) অ্যালগরিদম ব্যবহার করে মাল্টিভ্যারিয়েট (Multivariate) লিনিয়ার রিপ্রেশন	৫৯
পরিচেদ ৩.৬ : ভেক্টর অ্যালজেব্রা (Vector Algebra) ব্যবহার করে লিনিয়ার রিপ্রেশন	৬৪
পরিচেদ ৩.৭ : বায়াস (Bias), ভ্যারিয়েন্স (Variance) ও রেগুলারাইজেশন (Regularization)	৬৬
পরিচেদ ৩.৮ : গ্রেডিয়েন্ট ডিসেন্ট পদ্ধতির সাহায্যে লিনিয়ার রিপ্রেশনের উদাহরণ	৭৪
অধ্যায় ৪ : লজিস্টিক রিপ্রেশন (Logistic Regression)	৭৯

সূচি

পরিচেদ ৪.১ : কন্টিনিউয়াস ও ডিসক্রিট ডেটা (Continuous and Discrete Data)	৭৯
পরিচেদ ৪.২ : লজিস্টিক রিফ্রেশন (Logistic Regression)-এর হাইপোথিসিস	৮০
পরিচেদ ৪.৩ : লজিস্টিক রিফ্রেশনের কস্ট ফাংশন (Cost Function) ও প্রয়োগ পদ্ধতি	৮৪
পরিচেদ ৪.৪ : লজিস্টিক রিফ্রেশনে রেণুলারাইজেশনের ব্যবহার	৮৭
অধ্যায় ৫ : সাপোর্ট ভেস্ট্র মেশিন (Support Vector Machine - SVM)	৯০
পরিচেদ ৫.১ : লিনিয়ার সাপোর্ট ভেস্ট্র মেশিন (Linear Support Vector Machine - LSVM)	৯২
পরিচেদ ৫.২ : লিনিয়ার সাপোর্ট ভেস্ট্র মেশিনের একটি সহজ উদাহরণ	৯৭
পরিচেদ ৫.৩ : কার্নাল ট্রিক (Kernel Trick) ও নন-লিনিয়ার এসভিএম (Non-Linear SVM)	১০১
অধ্যায় ৬ : কে-নিয়ারেস্ট নেইবেরস (K-Nearest Neighbors)	১০৯
পরিচেদ ৬.১ : KNN-এর সাধারণ ধারণা	১১০
পরিচেদ ৬.২ : উদাহরণ	১১১
অধ্যায় ৭ : কে-মিনস ক্লাস্টারিং (K-Means Clustering)	১১৫
পরিচেদ ৭.১ : কে-মিনস ক্লাস্টারিংয়ের সংক্ষিপ্ত গাণিতিক বর্ণনা	১১৬
পরিচেদ ৭.২ : উদাহরণ	১১৮
অধ্যায় ৮ : নাইভ বেইজ ক্লাসিফায়ার (Naive Bayes Classifier)	১২৩
পরিচেদ ৮.১ : সন্তান্যতার টুকিটাকি	১২৪
পরিচেদ ৮.২ : বেইজ থিওরেম হাতে-কলমে	১২৫
পরিচেদ ৮.৩ : হাতে কলমে নাইভ বেইজ ক্লাসিফায়ার	১২৭
অধ্যায় ৯ : ডিসিশন ট্রি (Decision Tree)	১৩০
পরিচেদ ৯.১ : ডিসিশন ট্রি কী	১৩০
পরিচেদ ৯.২ : এন্ট্রোপি (Entropy)	১৩২
পরিচেদ ৯.৩ : গেইন (Gain)	১৩৫
পরিচেদ ৯.৪ : কীভাবে ডিসিশন ট্রি বানাব	১৩৭
অধ্যায় ১০ : প্রিসিপাল কম্পোনেন্ট অ্যানালাইসিস (Principal Component Analysis - PCA)	১৪১

সূচি

পরিচ্ছেদ ১০.১ : মিন (Mean), স্ট্যান্ডার্ড ডেভিয়েশন (Standard Deviation) এবং ভ্যারিয়েন্স (Variance) _____	১৪২
পরিচ্ছেদ ১০.২ : কোভ্যারিয়েন্স (Covariance) ও কোভ্যারিয়েন্স ম্যাট্রিক্স (Covariance Matrix) _____	১৪৩
পরিচ্ছেদ ১০.৩ : আইগেনভ্যালু (Eigenvalue) ও আইগেনভেক্টর (Eigenvectors) _____	১৪৫
পরিচ্ছেদ ১০.৪ : কীভাবে প্রিসিপাল কম্পোনেন্ট অ্যানালাইসিস করতে হয় _____	১৪৮
অধ্যায় ১১ : পারসেপ্ট্রন (Perceptron) _____	১৫৩
পরিচ্ছেদ ১১.১ : নিউরন (Neuron) এবং এর আদলে গাণিতিক কাঠামো _____	১৫৩
পরিচ্ছেদ ১১.২ : পারসেপ্ট্রন ট্রেনিং দেওয়া _____	১৫৬
অধ্যায় ১২ : একটুখানি নিউরাল নেটওয়ার্ক _____	১৬০
অধ্যায় ১৩ : পারফরম্যান্স (Performance) _____	১৬৩
পরিচ্ছেদ ১৩.১ : আর-স্কয়ারড ভ্যালু (R-Squared Value) _____	১৬৪
পরিচ্ছেদ ১৩.২ : কনফিউশন ম্যাট্রিক্স (Confusion Matrix) _____	১৬৫
পরিচ্ছেদ ১৩.৩ : অ্যাকুরেসি (Accuracy) _____	১৬৭
পরিচ্ছেদ ১৩.৪ : প্রিসিশন (Precision) ও রিকল (Recall) _____	১৬৮
পরিচ্ছেদ ১৩.৫ : এফ-ওয়ান মেজার (F1 Measure) _____	১৭১
পরিচ্ছেদ ১৩.৬ : স্পেসিফিসিটি (Specificity) _____	১৭৩
পরিচ্ছেদ ১৩.৭ : আরওসি কার্ভ (ROC Curve) _____	১৭৪
পরিচ্ছেদ ১৩.৮ : এলবো মেথড (Elbow Method) _____	১৭৯
অধ্যায় ১৪ : পরবর্তী সময়ে কী থাকছে _____	১৮২

অধ্যায় ১ : কৃতিম বুদ্ধিমত্তা এবং মেশিনের যত কথা

Imagination is more important than knowledge – বলেছিলেন বিজ্ঞানী আলবার্ট আইনস্টাইন।

চলুন, একটি কল্পনা দিয়ে শুরু করি। ধরা যাক, কোনো এক বিশ্ববিদ্যালয়ের কয়েকজন মেধাবী শিক্ষার্থী একটি রোবট তৈরি করেছে। রোবটটির বিশেষত্ত্ব হচ্ছে সে ভালো ফুটবল খেলতে পারে। রোবটটির নাম দেওয়া হয়েছে রোবু। রোবুর পায়ে যদি ফুটবল দেওয়া হয়, সে সুন্দর করে বলটি মাঠের এক প্রান্ত থেকে অন্য প্রান্তে টেনে নিয়ে গোল দিয়ে দিতে পারে। শুধু তা-ই নয়, রোবু অন্যান্য খেলোয়াড়ের সঙ্গে দু-তিনবার খেললেই তাদের খেলার ধরন কেমন সেটি মনে রাখতে পারে। এখন ধরি, ফিফা (FIFA) রোবু ও রোনালদো (Ronaldo)-এর মধ্যে একটি প্রীতি ফুটবল ম্যাচ সিরিজের আয়োজন করেছে। খেলার নিয়ম সব সময়কার মতোই। নির্দিষ্ট সময়সীমা বেঁধে দেওয়া থাকবে। সেই সময়সীমার মধ্যে যে সবচেয়ে বেশি পরিমাণ গোল দেবে, সে-ই ম্যাচ জিতবে। এরকম করে সর্বমোট ৫টি ম্যাচ খেলা হবে। ৫ ম্যাচ শেষ হওয়ার পর যে সবচেয়ে বেশি ম্যাচ জিতবে, সে সিরিজ-বিজয়ী বলে ঘোষিত হবে।

রোবু ও রোনালদোর মধ্যেকার সেই ঐতিহাসিক ৫ ম্যাচ সিরিজের ফলাফল টেবিল 1.1-এ দেওয়া হলো :

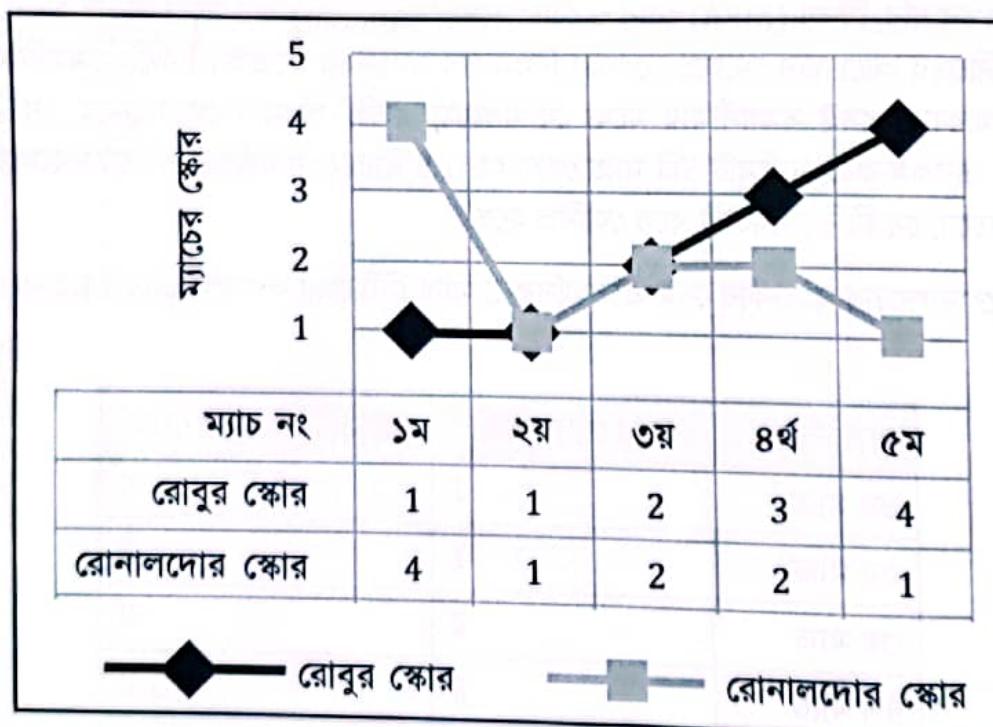
ম্যাচ নম্বর	রোবুর গোলসংখ্যা	রোনালদোর গোলসংখ্যা
১ম ম্যাচ	1	4
২য় ম্যাচ	1	1
৩য় ম্যাচ	2	2
৪র্থ ম্যাচ	3	2
৫ম ম্যাচ	4	1

টেবিল 1.1

আমরা যদি টেবিল 1.1 একটু ভালোভাবে লক্ষ করি, তাহলে দেখব যে, প্রথম ম্যাচে রোবু বিশাল ব্যবধানে হেরেছে, পরের দুটো ম্যাচ ড্র হয়েছে এবং শেষের দুই ম্যাচ রোবু জিতেছে। এই ফলাফলের ভিত্তিতে আমরা খুব সহজেই দাবি করতে পারি যে রোনালদোর বিপরীতে খেলতে খেলতে রোবু ক্রমান্বয়ে ভালোভাবে পারফরম করেছে। অর্থাৎ প্রতি ম্যাচে তার খেলা আগের

ম্যাচের চেয়ে উন্নত মানের হয়েছে শুধু আগের ম্যাচগুলোর অভিজ্ঞতার ওপর ভিত্তি করে। প্রতিটি ম্যাচ খেলার সময় সে ধীরে ধীরে রোনালদোর খেলার বিভিন্ন কলাকৌশল শিখে নিতে পেরেছে এবং পরবর্তী সময়ে সেগুলো ঠিকমতো কাজে লাগিয়ে সে বিজয় ছিনিয়ে এনেছে।

রোবু ও রোনালদোর মধ্যেকার 5 ম্যাচ সিরিজের ক্ষেত্রে 1.1-এ গ্রাফ আকারে দেখানো হলো। এই গ্রাফে বর্গাকৃতির মার্কারগুলো দিয়ে যে রেখা আঁকা হয়েছে, সেটি বোঝাচ্ছে রোনালদোর ক্ষেত্র; আর রহস্য বা ডায়মন্ড আকৃতির মার্কারগুলো দিয়ে যে রেখা আঁকা হয়েছে, সেটি বোঝাচ্ছে রোবুর ক্ষেত্র। এখানে হরাইজন্টাল (Horizontal) বা X-অক্ষ বরাবর নেওয়া হয়েছে ম্যাচসংখ্যা এবং ভার্টিকাল (Vertical) বা Y-অক্ষ বরাবর নেওয়া হয়েছে বিভিন্ন ম্যাচে রোবু আর রোনালদোর দেওয়া গোলের সংখ্যা। গ্রাফটিতে দেখা যাচ্ছে যে প্রথম থেকে পঞ্চম ম্যাচ পর্যন্ত প্রতি ম্যাচে রোবুর পারফরম্যান্স আগের ম্যাচের চেয়ে ক্রমান্বয়ে ভালো হয়েছে এবং এটি হয়েছে তার আগের ম্যাচগুলো থেকে শেখা অভিজ্ঞতার ওপর ভিত্তি করে।



ছবি 1.1 : রোবু ও রোনালদোর ম্যাচের ক্ষেত্র গ্রাফ

এখন এই ঘটনার ওপর ভিত্তি করে কি আমরা বলতে পারি যে রোবুর বুদ্ধি আছে? উত্তর হচ্ছে, হ্যাঁ। আমরা ধরে নিতে পারি যে রোবুর ভেতরে বুদ্ধিমত্তা আছে। যদিও এই বুদ্ধিমত্তা আর মানুষের বুদ্ধিমত্তার মধ্যে আকাশ-পাতাল ফারাক। রোবুর বুদ্ধিমত্তা খুবই নিচু মানের হলেও অন্তত কিছুটা বুদ্ধিমত্তা তার ভেতরে আছে।

অধ্যায় ১ : কৃতিম বুদ্ধিমত্তা এবং মেশিনের যত কথা

আমরা রোবুকে কেন বুদ্ধিমান বলতে পারছি? রোবুকে যদি আমরা একজন জলজ্যাস্ত মানুষের সঙ্গে তুলনা করি, তাহলে দেখব যে একজন বুদ্ধিমান মানুষের মতোই রোবু আচরণ করছে। কিংবা আরেকটু ভালোভাবে বলা যেতে পারে, রোবু একজন মানুষকে অনুকরণ করার চেষ্টা করছে। একজন সাধারণ মানুষ হলে কী করত? প্রতিবার একই প্রতিপক্ষের সঙ্গে ফুটবল খেলতে খেলতে কিছুটা হলেও তার পারফরম্যান্স আগের চেয়ে ভালো হতো। এর কারণ কী? কারণ বেশ কয়েকবার খেলার পরে সে অবশ্যই তার প্রতিপক্ষ সম্পর্কে কিছুটা হলেও জেনে ফেলবে। ফলে তার পারফরম্যান্স আগের চেয়ে ভালো হবে। তাই না? রোবুর ক্ষেত্রেও চিন্তা করলে দেখা যাবে, ঠিক তা-ই হয়েছে। আর তাই আমরা রোবুকে কিছুটা হলেও বুদ্ধিমান বলতে পারি।

তাহলে, কোনো প্রাণীকে যদি কোনো কাজ বেশ কয়েকবার করতে দেওয়া হয় এবং প্রতিবার তার কাজের ফলাফল আগের চেয়ে ভালো হয়ে উঠতে থাকে তার পূর্বের অভিজ্ঞতার কারণে তাহলে আমরা বলতে পারি, সেই প্রাণীর মধ্যে বুদ্ধিমত্তা রয়েছে। আর বুদ্ধিমত্তা যদি কৃতিমভাবে তৈরি করে দেওয়া যায় কোনো বস্তুর (কম্পিউটার, রোবট ইত্যাদি) ভেতরে, তাহলে আমরা তাদের বুদ্ধিমত্তাকে বলতে পারব কৃতিম বুদ্ধিমত্তা বা আর্টিফিশিয়াল ইন্টেলিজেন্স।

আর্টিফিশিয়াল ইন্টেলিজেন্স হচ্ছে কোনো মেশিনের ভেতরে কৃতিম উপায়ে বুদ্ধিমত্তা সঞ্চার/তৈরি করা, তাকে এমনভাবে প্রোগ্রাম করা, যাতে সে নিজে নিজে মানুষের মতোই কোনো কিছুকে বিশ্বেষণ করে কোনো কাজ করতে পারে। মানুষ যেভাবে চিন্তাভাবনা করে, কোনো মেশিন যত ভালোভাবে সেটি অনুকরণ করতে পারবে, আমরা সেই মেশিনকে তত বেশি আর্টিফিশিয়াল ইন্টেলিজেন্ট বলব।

যাঁরা কম্পিউটার নিয়ে পড়াশোনা করেছেন তাঁরা অনেকেই হয়তো আর্টিফিশিয়াল ইন্টেলিজেন্সের সঙ্গে সঙ্গে আরো একটি টার্ম প্রায়শই শুনে থাকবেন, সেটি হলো মেশিন লার্নিং। মেশিন লার্নিং – এই টার্মটি খুবই জনপ্রিয়। অনেকেই হয়তো এর অর্থ কী সেটি জানেন, অনেকে জানেন না; কিন্তু নাম শুনেছেন। আমি অনেককে দেখেছি, যাঁরা মনে করেন, এটি বোধহয় যন্ত্রপাতি কীভাবে কাজ করে এটি জানার জন্য কোনো কোর্স। আবার অনেকে মনে করেন, এটি বোধহয় অ্যাসেম্বলি ল্যাঙ্গুয়েজের সঙ্গে সম্পৃক্ত কোনো কিছু।

মেশিন লার্নিং হচ্ছে আর্টিফিশিয়াল ইন্টেলিজেন্সের একটি শাখা। খুব সহজ করে যদি বলতে যাই কিংবা খুঁজতে যাই, তাহলে কিন্তু আমরা নামের মধ্যেই অর্থ পেয়ে যেতে পারি। মেশিন লার্নিং হচ্ছে কোনো মেশিনকে লার্ন করানো, শেখানো। এখন প্রশ্ন হচ্ছে কী শেখাব? আর মেশিন বলতেই-বা কী বোঝানো হচ্ছে? উত্তর হলো, এখানে মেশিন হচ্ছে মূলত কম্পিউটার কিংবা রোবট আর মেশিনকে আমরা শেখাব কীভাবে আসলে নিজে নিজে চিন্তা করতে হয়।

মেশিন লার্নিং অ্যালগরিদম

সাধারণত আমরা যখন কম্পিউটার নিয়ে কোনো কাজ করি, সেটি সিনেমা দেখা হোক, গান শোনা হোক, গেম খেলা হোক, প্রতিটি জিনিস কিন্তু আমাদের কম্পিউটারকে বলে দিতে হয়, তাই না? আমরা যখন কোনো মূভি ফাইলের ওপরে ডাবল ক্লিক করি, তার মানে আমরা আসলে চাই কম্পিউটার যেন ওই মূভিটি আমাদের জন্য চালিয়ে দেয়। কম্পিউটার কিন্তু কখনোই নিজে থেকে বুঝে নেবে না যে আপনি এখন মূভি দেখার মুড়ে আছেন, তাই সে কখনোই অটোম্যাটিক মূভি চালিয়ে দেবে না (অস্তত এখন পর্যন্ত না, ভবিষ্যতে হলেও হতে পারে)। আপনাকে নিজে বসে মূভি ফাইল ক্লিক করে কম্পিউটারকে বলে দিতে হবে যে মূভিটি চালিয়ে দাও। আমাদের কম্পিউটারে করা অন্যান্য কাজও কিন্তু একইভাবে ঘটে, যদি আমরা একটু চিন্তা করে দেখি।

এখন, ব্যাপারটি কী রকম হবে যদি আপনি শুধু কম্পিউটারের সামনে বসে কম্পিউটারটি চালু করেন এবং সেই কম্পিউটার নিজে নিজে কতগুলো কাজ করে ফেলে –

- সে ওয়েবক্যাম দিয়ে আপনার চেহারার বেশ কিছু ছবি নেয়।
- সেই ছবি বিশ্লেষণ করে সে নিজেই নিজে বুঝে নেবে আপনার মন-মেজাজ এখন কী রকম।
- আপনার মন-মেজাজের ওপরে ভিত্তি করে সে নিজে নিজেই কোনো মূভি/গান চালিয়ে দেবে। আপনার মেজাজ খারাপ থাকলে কোনো হাসির মূভি চালাবে, যদি মেজাজ ভালো থাকে তাহলে হয়তো আপনাকে সায়েস্ফ ফিকশন বা কোনো থ্রিলার/অ্যাকশন মূভি চালিয়ে দিতে পারে।

একটু চিন্তা করে দেখুন তো, কেমন হবে এই ব্যাপারটা? যদি ধরতে পেরে থাকেন, আসলে কী ঘটল এখানে, তাহলে আপনি মেশিন লার্নিংয়ের মূল ভাবনাটি ধরতে পেরেছেন অনেকখানিই। আমাদের মেশিন লার্নিংয়ের মাধ্যমে লক্ষ্য এটাই থাকে, যেন আমরা কম্পিউটারকে একটু নিজে নিজে চিন্তা করতে শেখাতে পারি। যদি আমরা সাধারণভাবে চিন্তা করে দেখি, কম্পিউটার আসলে খুবই গাধা টাইপের একটি যন্ত্র। ওর কিন্তু নিজে নিজে কিছুই করার ক্ষমতা নেই। ওর ক্ষমতা একটিই, ওকে যদি কিছু বলে দেওয়া হয় সে সেটি খুব সুচারুভাবে অনেকক্ষণ ধরে কোনো রকম ভুল ছাড়া করে যেতে পারবে। এখন ওর এই গুণটির সঙ্গে যদি কিছু চিন্তাভাবনা করার ক্ষমতা জুড়ে দেওয়া যায়, তাহলে চিন্তা করতে পারেন, সেটি আমাদের জন্য কতখানি লাভজনক হবে?

এখন আমাদের এই ভয়াবহ রকমের গাধা টাইপ যন্ত্র কম্পিউটারকে যদি চিন্তাভাবনা করায় একটু হাতেখড়ি দিতে হয়, তাহলে অবশ্যই আমাদের মেশিন লার্নিংয়ের আশ্রয় নিতে হবে। মেশিন লার্নিংয়ের কিছু অ্যালগরিদম আছে, যেগুলো কম্পিউটারকে ভালোমতো শিখিয়ে-পড়িয়ে নিলে সে মোটামুটি একটি ছোটোখাটো বাচ্চা মুনি-খমির মতো ভবিষ্যৎ বলে দিতে পারবে। বিভিন্ন জিনিসপাতি দিলে একটি থেকে আরেকটি আলাদা করতে পারবে। একই ধরনের জিনিসগুলো বিভিন্ন প্যাটার্ন অ্যানালাইসিস করার মাধ্যমে একসঙ্গে গুচ্ছিয়ে ফেলতে পারবে।

অধ্যায় ১ : কৃতিম বৃক্ষিমত্তা এবং মেশিনের যত কথা

আরেকটি বিষয়, মেশিন লার্নিং ও আর্টিফিশিয়াল ইন্টেলিজেন্স – এই টার্ম দুটি প্রায় অনেক সহজেই একইসঙ্গে একে অন্যের পরিপূরক হিসেবে ব্যবহার করা হয়ে থাকে, যেটি আসলে পুরোপুরি সঠিক নয়। আর্টিফিশিয়াল ইন্টেলিজেন্স ও মেশিন লার্নিংয়ের মধ্যে বেশ অনেকটুকু পার্থক্য রয়েছে। মেশিন লার্নিংকে বলা যেতে পারে আর্টিফিশিয়াল ইন্টেলিজেন্স অর্জন করার একটি উপায়। আমাদের লক্ষ্য হচ্ছে মেশিনকে কোনো একটি কাজে ভালোভাবে পারফরম করতে শেখানো। সেটি কিন্তু আমরা একেবারে হাজার হাজার লাইনের হার্ডকোড করেও করতে পারতাম। সেটি না করে আমরা যেটি করি, অসংখ্য ডেটা দিয়ে দিই মেশিনকে এবং শিখিয়ে দিই যে কীভাবে তাকে ভেটার ভেতর থেকে নিজে নিজে শিখে নিতে হবে। এই পদ্ধতিটিই হচ্ছে মেশিন লার্নিং, যেটি শুধু আমাদের মেশিনগুলোকে আর্টিফিশিয়ালি ইন্টেলিজেন্ট করে তোলার একটি পদ্ধতি মাত্র।

আমি এই বইতে চেষ্টা করব মোটামুটি সবগুলো গুরুত্বপূর্ণ মেশিন লার্নিং অ্যালগরিদম দেখাতে। আমরা খুব বেশি কঠিন গাণিতিক হিসাবকিতাবে যাব না। যতটুকু দরকার, ঠিক ততটুকুই সহজ করে শিখব। আমাদের লক্ষ্য হবে প্রতিটি অ্যালগরিদমের বেসিক কনসেপ্টটুকু বুঝে নিয়ে এরপর এটি রিয়েল লাইফে/প্রোজেক্টে কোথায় এবং কীভাবে অ্যাপ্লাই করতে হয়, সেটি শেখাব।

এই বইতে যে অ্যালগরিদমগুলো দেওয়া আছে সেগুলো কোড করার জন্য মূলত ব্যবহার করা ভালো পাইথন। তাই যাঁদের পাইথনে দুর্বলতা আছে, কিংবা একেবারেই জানা নেই, তাঁরা মোটামুটি পাইথনের বেসিক Syntax-গুলো একটু দেখে নিতে পারেন ইন্টারনেটের বিভিন্ন রিসোর্স থেকে। কোডগুলো আমি নিজে এই বইতে করে দেব না, করতে হবে আপনাদের নিজেদেরই। যদি নিজে নিজে অ্যালগরিদম বুঝে মাথা খাটিয়ে করে ফেলতে পারেন, তাহলেই কেবল হাত পাকবে মেশিন লার্নিংয়ে।

কেউ যদি আমার বইয়ে দেওয়া মেশিন লার্নিংয়ের অ্যালগরিদমগুলোর কোড করে দেখতে চান, তাহলে পাইথন ডাউনলোড করে নিতে পারেন এই লিংক থেকে –

<https://www.python.org/downloads/>

সেই সঙ্গে আমরা পাইথনে মেশিন লার্নিংয়ের জন্য Scikit-Learn নামে একটি লাইব্রেরি ব্যবহার করি। কেউ চাইলে সেটি দেখার জন্য চু মেরে আসতে পারেন এই লিংকে –

<http://scikit-learn.org/stable/>

তবে আমি মনে করি পাইথন ও মেশিন লার্নিংয়ের জন্য প্রয়োজনীয় সব লাইব্রেরি একবারে ইন্স্টল করার সবচেয়ে সহজ উপায় হলো anaconda ইন্স্টল করে নেওয়া। এটি একবার নামিয়ে ইন্স্টল করে নিলে আর আলাদা করে আপনার কিছুই ইন্স্টল করা লাগবে না। এটি পাইথনের একটি ওপেন সোর্স ডিস্ট্রিবিউশন, যাতে পাইথন কম্পাইলারসহ পাইথনের আরো জনপ্রিয় লাইব্রেরিগুলো, যেমন – NumPy, Pandas, Scikit-Learn, Matplotlib, SciPy ইত্যাদি সব

মেশিন লার্নিং অ্যালগরিদম

আগে থেকেই ইনস্টল করা আছে। সেই সঙ্গে, এতে আছে Jupyter Notebook, যেটি পাইথন কোড গুছিয়ে লেখা এবং ডিজ্যুয়ালাইজেশনের জন্য খুবই জনপ্রিয় একটি অ্যাপ্লিকেশন। আপনারা anaconda নামিয়ে নিতে পারেন এই লিংক থেকে –

<https://www.anaconda.com/download/>

মেশিন লার্নিংয়ের দুনিয়ায় সবাইকে স্বাগত!

অধ্যায় ২ : বিভিন্ন ধরনের লার্নিং ও অন্যান্য

মেশিন লার্নিং – এই টার্মটি সবার আগে ব্যবহার করেন আর্থার স্যামুয়েল (Arthur Samuel, 1901-1990) নামের একজন আমেরিকান বিজ্ঞানী। তাঁকে আর্টিফিশিয়াল ইন্টেলিজেন্স ও কম্পিউটার গেমিংয়ের একজন পথিকৃৎ বলা চলে।



ছবি 2.1 : আর্থার স্যামুয়েল (Arthur Samuel, 1901-1990)

তাঁর তৈরি করা স্যামুয়েল চেকার্স প্লেয়িং প্রোগ্রামটিই বিশ্বের প্রথম প্রোগ্রাম, যা নিজে নিজে খেলা শিখতে পেরেছিল। স্যামুয়েল পড়াশোনা করেছিলেন ম্যাসাচুসেটস ইনসিটিউট অব টেকনোলজি (Massachusetts Institute of Technology – MIT) থেকে। পরবর্তীকালে তিনি বিশ্বখ্যাত বেল ল্যাবরেটরিতে কাজ করেছেন। এর মধ্যে বেশ কিছুদিন তিনি আইবিএম (IBM)-এ কাজ করেছেন। শেষ বয়সে এসে তিনি স্ট্যানফোর্ড বিশ্ববিদ্যালয় (Stanford University)-এ প্রফেসর হিসেবে যোগদান করেন। তিনি আইইই (IEEE) কম্পিউটার সোসাইটি থেকে কম্পিউটার পাইওনিয়ার অ্যাওয়ার্ড (Computer Pioneer Award)-সহ আরো অসংখ্য উল্লেখযোগ্য পুরস্কারে ভূষিত হয়েছেন। তাঁর মৃত্যু হয় পারকিনসন্স রোগের কারণে।

পরিচ্ছন্দ ২.১ : সুপারভাইজড ও আনসুপারভাইজড লার্নিং (Supervised and Unsupervised Learning)

মেশিন লার্নিং আলগরিদমগুলোকে সাধারণত দুটি ভাগে ভাগ করা যায় :

- Supervised Learning
- Unsupervised Learning

আরেকটি আছে Reinforcement Learning, সেটি আপাতত আমাদের এই বইয়ের এখতিয়ারের বাইরে থাকবে। ওটা নিয়ে পরবর্তী সময়ে অন্য কোনো সময় লেখা হবে।

আমরা এখন একটি সুপারভাইজড লার্নিং নিয়ে কথা বলি। সুপারভাইজ মানে হচ্ছে কাউকে কোনো কিছু করতে শিখিয়ে দেওয়া, দেখিয়ে দেওয়া – যাতে সে পরবর্তী সময়ে নিজে নিজে কাজটি করতে পারে। মেশিন লার্নিংয়ের ক্ষেত্রেও, সুপারভাইজড লার্নিং আলগরিদম হচ্ছে সেই সব অ্যালগরিদম, যেগুলো ব্যবহার করে আমরা কম্পিউটারকে মানুষের মত চিন্তা করা শেখানোর জন্য ট্রেনিং দেই। আর ট্রেনিং দেওয়ার জন্য আমাদের যেটি করতে হবে সেটি হচ্ছে কম্পিউটারকে পর্যাপ্তসংখ্যক উদাহরণ দিতে হবে।

একটি খুব সহজ উদাহরণ দিই। ধরা যাক, আপনি জীবনে কখনো জিরাফ দেখেননি। আমি যদি আপনাকে জিরাফ চেনাতে চাই, তাহলে কী করব? কী করাটা সবচেয়ে সহজ হবে? আমি, কিংবা আরো দশজন সাধারণ মানুষ যেটি করবে সেটি হলো আপনাকে চিড়িয়াখানায় নিয়ে গিয়ে জিরাফ দেখাবে, তাই না? জিরাফ দেখিয়ে বলবে, ‘দেখুন ভাই, এটি হলো জিরাফ।’ এর বিশাল লম্বা গলা থাকে। গায়ে হলুদ-বাদামি ফুটকি। এদের পাঞ্চলোও বেশ লম্বা। পেছনে লেজ আছে। মাথার ওপরে কান ছাড়াও দুটি ছোটো শিঙের মতো জিনিস আছে।’ তখন আপনি কী করবেন? আপনি ভালো করে জিরাফটি দেখবেন, ওর চেহারা আর ওর যে বৈশিষ্ট্যগুলো আছে, সেইগুলো মনে রাখার চেষ্টা করবেন।

এরপর আপনাকে কেউ যদি একটি প্রাণীর ছবি দেখিয়ে শনাক্ত করতে বলে যে, সেটি জিরাফ কি না, তখন কিন্তু আপনি অবশ্যই সেটি করতে পারবেন। কীভাবে? আপনি মনে মনে মিলিয়ে দেখবেন যে, এই নতুন প্রাণীর শরীরে জিরাফের বৈশিষ্ট্যগুলো আছে কি না। যদি থেকে থাকে, তখন আপনি বলবেন এটি জিরাফ, আর যদি না থাকে, তাহলে বলবেন এটি জিরাফ নয়।

আরেকটি উদাহরণ দিই। মনে করুন, আপনি পিংজা খেতে পছন্দ করেন। পিংজা হাটের পিংজা আপনার খুবই পছন্দের খাদ্য। এখন, যেহেতু আপনি নিয়মিতই সেখানে পিংজা খেতে যান, তাই পিংজা হাটের কিছু সাইজের পিংজার দাম আপনি আগে থেকেই জানেন (টেবিল 2.1-এ দেখানো হলো)।

অধ্যায় ২ : বিভিন্ন ধরনের লার্নিং ও অন্যান্য

পিংজার সাইজ (ইঞ্চিতে)	পিংজার দাম (টাকায়)
6"	399/-
9"	699/-
12"	945/-
15"	1215/-

টেবিল 2.1

এখন আপনার কোনো বন্ধু যদি আপনাকে জিজ্ঞাসা করে – ‘দোষ্ট, নতুন একটি পিংজা এসেছে শনলাম, 14 ইঞ্চি পিংজা। পুরো চিকেন আর চিজে মাখামাখি। পিংজার দাম কত হতে পারে আন্দাজ কর তো?’ তাহলে আপনি কিন্তু খুব সহজেই আন্দাজ করে বলতে পারবেন যে 1150/- টাকার মতো হবে দাম। এটি কীভাবে বলবেন?

আপনি নিজের মনের অজান্তেই চিন্তা করবেন যে 15 ইঞ্চি পিংজার দাম আপনি জানেন 1215/- টাকা, আর 12 ইঞ্চি পিংজার দামও আপনি জানেন 945/- টাকা। তার মানে 14 ইঞ্চি পিংজার দাম এ দুটির মাঝামাঝি কিছু একটা হবে। যেহেতু 14 ইঞ্চি পিংজা আর 15 ইঞ্চি পিংজার সাইজ বেশ কাছাকাছি, সুতরাং এদের দামও কাছাকাছিই হবে। তাই 1150/- টাকার কাছাকাছি কিছু একটা হবে দাম। মজার ব্যাপার হচ্ছে, এই চিন্তাটি আপনি করবেন আপনার মনের অজান্তেই এবং এত দ্রুত করবেন যে আপনি টেরই পাবেন না কীভাবে আপনি দাম আন্দাজ করলেন। একইভাবে আপনাকে যদি 16 ইঞ্চি পিংজার দাম আন্দাজ করতে বলা হয় আপনি সেটিও করতে পারবেন (এই উদাহরণটি নিয়ে আমরা সামনে আরো বিস্তারিত আলোচনা করব)।

ওপরের যে দুটি উদাহরণ দেখানো হলো, দুটিই সুপারভাইজড লার্নিংয়ের উদাহরণ। প্রথম উদাহরণে প্রথমে জিরাফ দেখিয়ে আর জিরাফের বৈশিষ্ট্য বলে আপনার মন্ত্রিককে ট্রেনিং দেওয়া হলো জিরাফ চেনার জন্য। এরপর আপনাকে যত প্রাণীর ছবিই দেওয়া হোক না কেন, আপনি আলাদা করতে পারবেন যে কোনটি জিরাফ আর কোনটি জিরাফ নয়। এ ধরনের সমস্যাগুলোকে আমরা বলি ক্লাসিফিকেশন প্রবলেম (Classification Problem)।

আবার দ্বিতীয় উদাহরণে আপনাকে প্রথমে বিভিন্ন পিংজার সাইজ ও তাদের দাম বলে দেওয়া হলো, আর তারপরে সেই তথ্যের ওপরে ভিত্তি করে আপনাকে আন্দাজ করতে বলা হলো অন্য আরেক সাইজের পিংজার দাম যেটি আপনার অজানা। এ ধরনের সমস্যাগুলোকে আমরা বলি রিপ্রেশন প্রবলেম (Regression Problem)।

আমরা রিপ্রেশন ও ক্লাসিফিকেশন, দুই ধরনের প্রবলেম নিয়েই সামনে বিস্তারিত আলোচনা করব এবং এইসব সমস্যা সমাধান করার জন্য বিভিন্ন অ্যালগরিদম দেখব।

এবার আসি আনসুপারভাইজড লার্নিংয়ে। সুপারভাইজড লার্নিংয়ে আমরা যেরকম ডেটার পাশাপাশি সঠিক উত্তরটিও দিয়ে দিতাম, আনসুপারভাইজড লার্নিংয়ে সেরকম ধরনের কোনো ট্রেনিং দিয়ে নেওয়া হবে না কম্পিউটারকে। তাকে ট্রেনিং-এর জন্য শুধু ডেটা দিয়ে দেওয়া হবে, সঠিক উত্তর নয়। সে নিজের মতো করে চেষ্টা করবে ওই ডেটার মধ্যে প্যাটার্ন বা বিভিন্ন ডেটার মধ্যে সামঞ্জস্য খুঁজে বের করতে। সেই সামঞ্জস্যের ওপরে ভিত্তি করে সে সবগুলো ডেটা বিভিন্ন গ্রুপে সাজাবে। এই ধরনের সমস্যাগুলোকে আমরা বলি ক্লাস্টারিং প্রবলেম (Clustering Problem), যা কিনা আনসুপারভাইজড লার্নিংয়ের সবচেয়ে জনপ্রিয় উদাহরণ। কিন্তু এর বাইরেও আরো বেশ কিছু ধরনের সমস্যা আছে আনসুপারভাইজড লার্নিংয়ের ক্ষেত্রে। আমাদের এই বইতে আমরা অবশ্য ক্লাস্টারিং প্রবলেম নিয়েই বিষদ আলোচনা করব।

পরিচ্ছেদ ২.২ : ফিচার (Feature)

এখন আমাদের আরো কিছু মেশিন লার্নিংয়ের ধারণার ব্যাপারে একটু জেনে নিতে হবে। এগুলো একেবারে গোড়ার দিকের কিছু ধারণা এবং এগুলো মেশিন লার্নিং শুরু করার জন্য জানা খুবই দরকার। এই ধারণাগুলো যদি পরিষ্কার না থাকে, তাহলে পরবর্তী সময়ে বাকি বিষয়গুলো বুঝতে অসুবিধা হবে।

টেবিল 2.2 ভালো করে লক্ষ করি। এই টেবিলে কয়েকটি ভিন্ন ভিন্ন প্রাণীর ছবি থেকে ওই প্রাণী সম্পর্কে কিছু ভিন্ন ভিন্ন ডেটা দেওয়া আছে। এই টেবিল ব্যবহার করে আমরা আমাদের পরবর্তী কনসেপ্টগুলো শিখব। ডেটাটি কাল্পনিক, বুঝতেই পারছেন। শুধু আপনাদের বোঝানোর উদ্দেশ্যে ব্যবহার করা হচ্ছে। এর সঙ্গে কেউ বাস্তবের সামঞ্জস্য খুঁজতে যাবেন না যেন।

ছবি নম্বর	লেজের দৈর্ঘ্য (সে.মি.)	গলার দৈর্ঘ্য (সে.মি.)	শিঙের মতো বস্তু কি আছে?	প্রাণীটি কি জিরাফ?
1	5	8	হ্যাঁ	হ্যাঁ
2	2	3	না	না
3	1	2	না	না
4	0	2	না	না
5	5.5	7.5	হ্যাঁ	হ্যাঁ

টেবিল 2.2

অধ্যায় ২ : বিভিন্ন ধরনের লার্নিং ও অন্যান্য

একটি বিষয় চিন্তা করে দেখুন, আমরা যখন কাউকে নতুন কোনো কিছু চেনাই, তখন তাকে সেই বন্টির সবচেয়ে অনন্য বা ইউনিক (unique) বৈশিষ্ট্যগুলোর কথা বলি, যা দিয়ে সে সহজেই তাকে আলাদা করতে পারবে আর দশটি অন্য কিংবা একই রকম জিনিস থেকে।

যেমন ধরা যাক, আপনার ক্লাসে দুজন আছে আরিফ নামে। একজন চশমা পরে, আরেকজন পরে না। এখন, একদিন ক্লাসে গিয়ে আপনি আরিফকে খুঁজছেন কোনো একটি দরকারে। তাকে ক্লাসে না পেয়ে আপনি আপনার কোনো এক বন্ধুকে জিগ্যেস করলেন, ‘দোষ্ট, আরিফ কোথায় রে?’ তখন সে স্বাভাবিকভাবেই আপনাকে জিগ্যেস করবে, ‘কোন আরিফ? ক্লাসে তো দুজন আরিফ আছে।’ তখন আপনি বলবেন হয় চশমা-পরা আরিফ, কিংবা চশমা-ছাড়া আরিফ।

এই যে আপনি দুজন আরিফকে তাদের একটি বৈশিষ্ট্য দিয়ে আলাদা করলেন, এই বৈশিষ্ট্যটিই হলো ফিচার (Feature)। তার মানে ফিচার হচ্ছে সেই সমস্ত অনন্য বৈশিষ্ট্য, যা কোনো কিছুকে আর দশটি সামঞ্জস্যপূর্ণ জিনিস থেকেও সহজে আলাদা করে ফেলতে পারে।

এখন, আমরা যদি কম্পিউটারকে কোনো কিছু চেনাতে চাই, তাহলে কী করব? ধরুন, আমরা যদি কম্পিউটারকে একটি জিরাফ চেনাতে চাই, তাহলে, প্রথমে আমরা কী করব? কম্পিউটারকে ট্রেনিং দেব। কীভাবে ট্রেনিং দেব? তাকে বিভিন্ন প্রাণীর ছবি ইনপুট হিসেবে দেব এবং সেই সঙ্গে সেই সব প্রাণীর লেজের দৈর্ঘ্য, গলার দৈর্ঘ্য, মাথার ওপরে শিং আছে কি নেই – এসব ডেটাও ইনপুট হিসেবে দেব (টেবিল 2.2-এর মতো)। অথবা তাদের এমনভাবে নির্দেশ দেব, যাতে তারা সেই ফিচারগুলো নিজেরাই ছবি থেকে বিভিন্ন অ্যালগরিদম ব্যবহার করে বের করে নিতে পারে।

এখন ধরা যাক, যদি সুপারভাইজড লার্নিং হয়, তাহলে তাদেরকে এইসব ফিচার ডেটার সঙ্গে সঙ্গে এটি ও বলে দেওয়া হবে কোনটি জিরাফ, আর কোনটি জিরাফ নয়। তখন, আমাদের ডেটাকে বলা হবে লেবেলড ডেটা (Labelled Data)। লেবেলড ডেটাতে আউটপুটের প্রকৃত মান বলে দেওয়া থাকে। টেবিল 2.2 থেকে যদি বলতে যাই, শেষের কলামটি হচ্ছে প্রাণীর লেবেল (অর্থাৎ সে জিরাফ, নাকি জিরাফ নয়), আর তাঁর আগের যাবতীয় সব কলাম হচ্ছে ফিচার ডেটা।

কিন্তু, যদি আনসুপারভাইজড লার্নিং হয়, তাহলে আর কম্পিউটারকে বলে দেওয়া হবে না যে, কোনটি জিরাফ আর কোনটি নয়। কম্পিউটার নিজে নিজেই ফিচার ডেটার মধ্যে একটি সামঞ্জস্য বের করার চেষ্টা করবে। তখন আমাদের ডেটাকে বলা হবে আনলেবেলড ডেটা (Unlabelled Data) অর্থাৎ ডেটাতে আউটপুটের প্রকৃত মান বলে দেওয়া থাকবে না। টেবিল 2.2 থেকে যদি বলতে যাই, টেবিলের শেষের কলামটি দেওয়া না থাকলেই এটি হয়ে যাবে আনলেবেলড ডেটা।

এইসব ডেটা (লেবেলড/আনলেবেলড) কম্পিউটার তার ডেটাবেজে সংরক্ষণ করে রাখবে। এরপরে কম্পিউটারকে আমরা নতুন কতগুলো ভিন্ন ভিন্ন প্রাণীর ছবি দেব, যে ছবিগুলো আগে সে

দেখেনি। সে ছবিগুলোর মধ্যে কোনগুলো জিরাফ আর কোনগুলো জিরাফ নয়, কম্পিউটার সেটি চিহ্নিত করবে।

এটি করার জন্য সে প্রথমে যা করবে, সেটি হলো নতুন ছবির প্রাণীটির গলা, লেজ, মাথা – এগুলো শনাক্ত করবে। তারপরে গলার দৈর্ঘ্য (নিউমেরিক ডেটা), লেজের দৈর্ঘ্য (নিউমেরিক ডেটা), মাথায় শিখের মতো বস্তু আছে কি নেই (হ্যাঁ/না, বাইনারি ডেটা হতে পারে) ইত্যাদি বের করে নেবে।

এরপরে নিজের ডেটাবেজে থাকা তথ্যের সঙ্গে এই নতুন পাওয়া তথ্য মিলিয়ে দেখবে যে দুটো মোটামুটি কাছাকাছি হয় কি না। যদি হয়, তাহলে ছবিটাকে জিরাফ বলে শনাক্ত করবে, নচেৎ নয়।

আগেই বলেছি, আমাদের কাছে থাকা সমস্ত ডেটার মধ্যে যেসব ডেটা কম্পিউটারকে সাহায্য করছে কোনটি জিরাফ আর কোনটি জিরাফ নয় সেটি শনাক্ত করতে (লেজের দৈর্ঘ্য, গলার দৈর্ঘ্য, শিখের উপস্থিতি ইত্যাদি), সেগুলোকেই আমরা বলব ফিচার। আর ইনপুট থেকে ফিচার ডেটা হিসাব করে বের করে নেওয়াকে বলে ফিচার এক্সট্রাকশন (Feature Extraction)। তার মানে, টেবিল 2.2 থেকে যদি বলি, লেজের দৈর্ঘ্য, গলার দৈর্ঘ্য এবং শিখের মতো বস্তুর উপস্থিতি – এই তিনটি তথ্য ফিচার ডেটা হিসেবে কাজ করবে।

ফিচার ডেটা নিয়ে হিসাব করে, অ্যানালাইসিস করে কম্পিউটার যে ফলাফল দেবে সেটাই হবে আমাদের কাঞ্চিত আউটপুট। এই আউটপুটকে আবার রেসপন্স ভ্যারিয়েবল (Response variable)-ও বলে।

কম্পিউটারকে ভিন্ন ভিন্ন কতগুলো প্রাণীর ছবি দিয়ে সেগুলোর মধ্যে কোনগুলো জিরাফ আর কোনগুলো জিরাফ নয়, সেগুলো শনাক্ত করতে দিলে কম্পিউটার বিভিন্ন অ্যানালাইসিস করে প্রতিটি ছবিটি জিরাফের হলে ক্রিনে 'হ্যাঁ' লেখা প্রিন্ট করবে, আর জিরাফ না হলে 'না' লেখা প্রিন্ট করবে। এটাই হলো রেসপন্স ভ্যারিয়েবল বা আউটপুট। যেমন টেবিল 2.2-এ, একেবারের ডান কলামের মানগুলো হলো রেসপন্স ভ্যারিয়েবলের মান।

তাহলে রেসপন্স ভ্যারিয়েবল আর লেবেল-এর মধ্যে পার্থক্য কী? রেসপন্স ভ্যারিয়েবল হচ্ছে আমার অ্যালগরিদম যে মান আউটপুট হিসেবে দিচ্ছে সেটি। আর লেবেল হচ্ছে সেই মান দ্বারা আসলে কী বোঝানো হচ্ছে সেটি।

ধরা যাক, কম্পিউটার যদি জিরাফের ছবি পায়, তাহলে সে 1 আউটপুট দেবে, আর না পেলে 0 আউটপুট দেবে। এই 1 ও 0 হচ্ছে রেসপন্স ভ্যারিয়েবল যেটি অ্যালগরিদম আমাকে সরাসরি দিচ্ছে। আর 1 মানে হচ্ছে 'জিরাফ', এখানে এই জিরাফটি হলো লেবেল।

একইভাবে 0 মানে হচ্ছে 'জিরাফ নয়', এখানে 'জিরাফ নয়' এটি হচ্ছে লেবেল।

পরিচ্ছেদ ২.৩ : ট্রেনিং, টেস্ট ও ভ্যালিডেশন ডেটা (Training, Test and Validation Data)

যে ডেটা দিয়ে আমরা কম্পিউটারকে ট্রেনিং দেব, সে ডেটাকে বলে 'ট্রেনিং ডেটা (Training Data)'। আর যে ডেটা দিয়ে আমরা কম্পিউটারের ট্রেনিং শেষ হওয়ার পর তার পারফরম্যান্স অ্যানালাইসিস করব যে তার কাজে সে কতটুকু ভালো করছে, সেই ডেটাকে আমরা বলব 'টেস্ট ডেটা (Test Data)'। সাধারণত পুরো ডেটাসেটকে দুই ভাগে ভাগ করে একটি ভাগকে ট্রেনিং ডেটা, অন্যটিকে টেস্ট ডেটা এভাবে ব্যবহার করা হয় (Validation Data-এর ব্যাপারে পরে বর্ণনা করছি)। যেমন, আমার কাছে 100 ডেটা থাকলে 60টি ডেটা দিয়ে আমি ট্রেনিং দিতে পারি এবং বাকি 40টি ডেটা দিয়ে আমি টেস্ট করতে পারি। এই অনুপাতটি আমার কাজের প্রকারভেদের ওপরে নির্ভর করবে। কোনটি ট্রেনিং সেটে যাবে, আর কোনটি টেস্ট সেটে, এগুলো দৈবভাবে বা র্যানডমলি নির্বাচন করা হয়।

এই ট্রেনিং আর টেস্ট ডেটার ব্যাপারটি আরেকটু খোলাসা করে বলি। ধরা যাক, সুপারভাইজড লার্নিং নিয়ে আমরা কাজ করছি। তার মানে আমাদের ডেটা অবশ্যই হবে লেবেলড ডেটা, অর্থাৎ, রেসপন্স ভ্যারিয়েবলের আসল মান কিংবা সোজা কথায় আউটপুট কী হবে তা আমাদের আগে থেকেই জানা। এখন ধরা যাক, আমাদের কাছে 100টি ছবির লেবেলড ডেটা আছে। এর মধ্যে 50টি আমাদের ট্রেনিং ডেটা, আর বাকি 50টি টেস্ট ডেটা।

এখন ট্রেনিং ডেটা দিয়ে কম্পিউটার মূলত যা করবে সেটি হলো, সে মনে রাখার চেষ্টা করবে ফিচার ডেটার কোনটির মান ঠিক কী রকম হলে সেটি জিরাফ হচ্ছে, আর কী রকম হলে সেটি জিরাফ বাদে অন্য প্রাণী হচ্ছে। এরপর তাকে যখন টেস্ট ডেটা দেওয়া হবে, তখন সে সেই টেস্ট ডেটার ওপরে 'ফিচার ডেটার কোনটির মান ঠিক কী রকম হলে সেটি জিরাফ হচ্ছে, আর কী রকম হলে সেটি জিরাফ বাদে অন্য প্রাণী হচ্ছে' – এই জ্ঞানটুকু, যা সে ট্রেনিং ডেটা অ্যানালাইসিস করে পেয়েছে, প্রয়োগ করে দেখবে। অ্যানালাইসিস প্রয়োগ করার পরে সে যে রেসপন্স ভ্যারিয়েবল তৈরি করবে সেটি যদি টেস্ট ডেটার অরিজিনাল রেসপন্স ভ্যারিয়েবলের মানের সঙ্গে মিলে যায়, তাহলেই বুঝব আমাদের মেশিন শিখতে পেরেছে।

দুর্বোধ্য মনে হচ্ছে, তাই না?

তাহলে সময় নষ্ট না করে উদাহরণে যাই। টেবিল 2.2 দিয়েই বর্ণনা করি। টেবিলে দেখা যাচ্ছে, 5টি ছবির ডেটা দেওয়া আছে। আগেই বলেছি, আমরা সুপারভাইজড লার্নিং ব্যবহার করছি। তাই, আমাদের ডেটা হবে লেবেলড ডেটা, মানে রেসপন্স ভ্যারিয়েবলের আসল ভ্যালু দেওয়া থাকবে (ডান দিকের কলাম)।

এখন ধরা যাক, এই ৫টি ডেটার মধ্যে প্রথম ৩টি আমাদের ট্রেনিং, পরের ২টি টেস্ট ডেটা। ধরি, প্রথম ৩টি ট্রেনিং ডেটা থেকে কম্পিউটার হিসাবকিতাব করে বের করল যে লেজের দৈর্ঘ্য ৫ সেমি বা এর কাছাকাছি হলে, গলার দৈর্ঘ্য ৪ সেমি বা এর কাছাকাছি হলে এবং শিঙের মতো বন্ধ উপস্থিত থাকলে সেই প্রাণী জিরাফ, আর নাহলে সেটি অন্য প্রাণী। জিরাফ হতে হলে এই ৩টি রিকোয়ারমেন্ট/কন্ডিশনই সত্যি হতে হবে।

এখন যদি কম্পিউটারকে টেস্ট ডেটা দেওয়া হয় (৪ ও ৫ নম্বর ছবির ডেটা) তাহলে কম্পিউটার কী আউটপুট দেবে, বলতে পারবে? ৪ নম্বর ছবির জন্য কম্পিউটার অবশ্যই ‘না’ আউটপুট দেবে, কারণ শিঙের মতো বন্ধ নেই, আর গলা-লেজের দৈর্ঘ্যও আমাদের রিকোয়ারমেন্টের কাছাকাছি নেই। কিন্তু ৫ নম্বর ছবির জন্য কম্পিউটার অবশ্যই ‘হ্যাঁ’ আউটপুট দেবে, কেননা এটি আমাদের জিরাফ হওয়ার ৩টি রিকোয়ারমেন্টই পূরণ করে।

এখন তাহলে ৪ ও ৫-এর জন্য কম্পিউটার আউটপুট দেবে যথাক্রমে ‘না’ ও ‘হ্যাঁ’ – যা কিনা আমাদের কাছে থাকা অরিজিনাল আউটপুট ভ্যালুর সঙ্গে মিলে যাচ্ছে (চাটে মিলিয়ে দেখুন)।

সুতরাং, আমাদের কম্পিউটারের শেখার একিউরেসি আমরা 100% বলতে পারি এই ক্ষেত্রে (বাস্তবে এটি হয় না বললেই চলে কারণ আমরা বাস্তবে আরো অনেক ডেটা নিয়ে কাজ করি, ফলাফল ভুল হওয়ার আশঙ্কাও অনেক বেশি থাকে সে ক্ষেত্রে)।

একটি কথা খুব ভালোমতো বোঝার চেষ্টা করতে হবে, টেস্ট ডেটা নিয়ে কম্পিউটার যখন কাজ করে, তখন সে কিন্তু যেই রেসপন্স ভ্যারিয়েবলের ভ্যালু দেওয়া আছে, সেইটাই আউটপুট হিসেবে দিয়ে দেয় না। সে ট্রেনিং ডেটা থেকে পাওয়া রিকোয়ারমেন্ট অনুযায়ী অ্যানালাইসিস করে প্রতিটি টেস্ট ডেটার জন্য একটি করে আউটপুট দেয়।

এরপর সেটি আমাদের কাছে থাকা টেস্ট ডেটার অরিজিনাল আউটপুটের সঙ্গে মিলিয়ে দেখা হয় যে কয়টি মিলল আর কয়টি ভুল করল কম্পিউটার। এরপর সেই হিসাবে তখন একিউরেসি মাপা হয়, যেভাবে আমরা একটু আগে করে দেখালাম।

যা-ই হোক, এতক্ষণ আমরা দেখলাম ট্রেনিং ডেটা ও টেস্ট ডেটার ধারণাটুকু। এখান থেকে এতটুকু বোঝা যাচ্ছে যে, মডেল একেবারে নিজেকে ট্রেনিং দিয়ে পুরোপুরি তৈরি করে না ফেলা পর্যন্ত টেস্ট ডেটার দেখা পায় না, তাই না?

আমরা যদি আমাদের নিজেদের জীবনের সঙ্গে মেলাতে যাই, ধরন আপনি কোনো পরীক্ষা (ধরন GRE পরীক্ষা) দেবেন। পরীক্ষা দেওয়ার আগে আপনি অবশ্যই দু-তিন মাস খুব ভালোমতো পড়াশোনা করবেন, নিজেকে প্রস্তুত করবেন, তাই না? কী কী ধরনের প্রশ্ন আসতে পারে, সব ধরনের প্রশ্ন যাচাইবাছাই করে, সমস্ত টপিক ঠিকমতো শেষ করে নিজেকে প্রস্তুত করে তবেই তো পরীক্ষা দিতে যাবেন।

অধ্যায় ২ : বিভিন্ন ধরনের লার্নিং ও অন্যান্য

নিজেকে এই যে প্রস্তুত করার ব্যাপারটি হলো অনেকটা ট্রেনিং ডেটা দিয়ে নিজেকে ট্রেনিং দেওয়ার মতো।

আর যখন সমস্ত প্রস্তুতি শেষে পরীক্ষা দিতে যাবেন এবং পরীক্ষার রেজাল্ট আপনাকে বলে দেবে আপনার ক্ষেত্রে কত হয়েছে, সেটাই কিন্তু বলে দেবে আপনার প্রস্তুতি আসলেই কতটুকু ভালো ছিল। পরীক্ষার প্রশ্নই হচ্ছে আপনার টেস্ট ডেটা। আপনি কি পরীক্ষার প্রশ্ন কখনো আগেভাগে দেখে ফেলতে পারেন? কখনোই না, তাই না? ঠিক সেরকম, টেস্ট ডেটাও কখনো মডেলকে দেখতে দেওয়া হয় না। আগে সে সম্পূর্ণ প্রস্তুত হয়, এর পরে টেস্ট ডেটা হাতে পায়।

সবশেষে বলি, ভ্যালিডেশন ডেটার কথা। ভ্যালিডেশন ডেটা আপনার ট্রেনিং ডেটারই একটি অংশ, যেটি আপনার মডেল কতটুকু ভালো হয়েছে, আরো কতটুকু ভালো করা দরকার, মডেলের বিভিন্ন প্যারামিটারগুলো টিউন করার কাজ ইত্যাদি করতে সাহায্য করে।

এটি অনেকটা মূল পরীক্ষার পূর্বে, পরীক্ষার প্রস্তুতি হিসেবে মডেল টেস্ট দেওয়ার মতো। আপনি পড়াশোনা করে নিজেকে প্রস্তুত করলেন মূল পরীক্ষার জন্য (ট্রেনিং ডেটা), এরপর মূল পরীক্ষার আগে নিজের দুর্বলতা ঝালাই করে নেওয়ার জন্য মূল পরীক্ষার প্রশ্নের আদলেই কয়েকটি মডেল টেস্ট দিলেন (ভ্যালিডেশন ডেটা) এবং সেখান থেকে নিজের দুর্বলতা সব বের করে নিজেকে একেবারে প্রস্তুত করে তবেই ফাইনাল পরীক্ষা দিতে গেলেন (টেস্ট ডেটা)।

অর্থাৎ বলা চলে, ভ্যালিডেশন ডেটা হচ্ছে টেস্ট ডেটার মতোই এক ধরনের ডেটা, যা মডেল আগেভাগে ট্রেনিংয়ের সময় দেখতে পারে না। ট্রেনিং শেষ হওয়ার পরে নিজেকে কিছুটা যাচাই করার জন্য এটি ব্যবহার করে নিজেকে ঠিকমতো ‘টিউন’ করে নিতে পারে, যাতে সে টেস্ট ডেটার ওপরে ভালো ফলাফল করে। মূলত মডেলকে ভালো পারফরম করতে ঠিকমতো আপডেট করাই হচ্ছে এই ভ্যালিডেশন ডেটার উদ্দেশ্য।

আমরা তাহলে যেটি করতে পারি, আমাদের গোটা ডেটাসেটকে এখন তিন ভাগ করতে পারি –

- ট্রেনিং ডেটা,
- ভ্যালিডেশন ডেটা ও
- টেস্ট ডেটা

ধরুন আপনার কাছে যদি 100টি ডেটা থাকে, আপনি 60টি দিয়ে ট্রেনিং দিলেন, 20টি দিয়ে ভ্যালিডেশন করলেন এবং বাকি 20টি একেবারে সব প্রস্তুতি শেষে টেস্ট করার জন্য রেখে দিলেন।

মোটামুটি এই হচ্ছে ট্রেনিং, ভ্যালিডেশন ও টেস্ট ডেটার ধারণা। আশা করি, সবাই বুঝতে পেরেছেন।

পরিচেদ ২.৪ : ক্রস ভ্যালিডেশন (Cross Validation)

ক্রস ভ্যালিডেশন (Cross Validation) মেশিন লার্নিংয়ের অত্যন্ত গুরুত্বপূর্ণ একটি ধারণা। এই সঙ্গে কেউ কিন্তু আবার ভ্যালিডেশন ডেটাকে গুলিয়ে ফেলবেন না। দুটি একেবারে ভিন্ন ধারণা। এটির বিস্তারিত উদাহরণে যাওয়ার আগে দুটো ছোটো গল্প বলি।

ধরা যাক, ইশতিয়াক স্কুলে পড়ে। সামনে তার অঙ্ক পরীক্ষা। অঙ্ক বইতে মোট অধ্যায় আছে 10টি। পরীক্ষার আগে স্যার বলে দিলেন যে প্রথম 5 অধ্যায়ের ওপরে পরীক্ষা হবে, তাই ইশতিয়াক বইয়ের প্রথম 5 অধ্যায় পড়ে পরীক্ষা দিতে গেল। এখন ইশতিয়াক যদি পরীক্ষায় দিতে গিয়ে দেখে, পুরো বই থেকেই প্রশ্ন হয়েছে, শুধু প্রথম 5 অধ্যায় থেকে নয়; তাহলে ইশতিয়াক পরীক্ষার খারাপ করবে। তাই না? এখন, তার এই পরীক্ষায় খারাপ করার ওপরে ভিত্তি করে কেউ যদি বলে যে ইশতিয়াক খারাপ ছাত্র, সে পড়াশোনা করে না একদমই, তাহলে কি ঠিক হবে?

আবার ধরা যাক, ইশতিয়াকের স্যার বললেন, পরীক্ষা হবে পুরো বইয়ের ওপরে, অর্ধাং 10টি অধ্যায়ই সিলেবাসে থাকবে। ইশতিয়াক এবারে পুরো বইটা পড়ে পরীক্ষার হলে গিয়ে দেখল, শুধু প্রথম 5 অধ্যায় থেকেই সব প্রশ্ন এসেছে, পরের 5 অধ্যায় থেকে কিছুই আসেনি। এই পরীক্ষার ইশতিয়াক খুব ভালো নহুন পেল। কিন্তু, এই পরীক্ষায় কি ইশতিয়াকের সঠিক মূল্যায়ন হলো? আমরা কি এই শুধু ফলাফলের ওপর ভিত্তি করেই তাকে ভালো ছাত্র বলতে পারব?

ওপরের দুটো ক্ষেত্রে কোনোটিই আমরা সঠিকভাবে ইশতিয়াককে মূল্যায়ন করিনি। সঠিকভাবে মূল্যায়ন তখনই হতো, যদি যতগুলো অধ্যায় পরীক্ষায় দেওয়া হয়েছে সবগুলো থেকেই ইশতিয়াককে প্রশ্ন করা হতো এবং যতটুকু সিলেবাস তার মধ্যে থেকেই প্রশ্ন করা হতো।

এখন আসি ক্রস ভ্যালিডেশনের ক্ষেত্রে। আমরা ইতিমধ্যেই দেখেছি যে, আমরা যে ডেটাসেট নিই, তার কিছু অংশ আমরা ট্রেনিং ডেটা হিসেবে ব্যবহার করি, আর বাকি অংশ টেস্ট ডেটা হিসেবে। এখন, ধরা যাক, আমার কাছে 100টি ডেটা পয়েন্ট আছে। এই ডেটা সেটকে, 5টি সমান ভাগে ভাগ করব। তাহলে আমার প্রতি ভাগে ডেটা পয়েন্ট থাকবে 20টি করে, ঠিক? টেবিল 2.4.1-এর দিকে তাকালে বোধ যাবে ভালোমতো। আমাদের ডেটাসেটের এই পাঁচটি সাবসেটকে আমরা যথাক্রমে D1, D2, D3, D4 ও D5 নাম দিয়ে চিহ্নিত করে দিই।

D1	D2	D3	D4	D5
20	20	20	20	20

টেবিল: 2.4.1

এখন আমরা যদি একক্ষণ যেভাবে ট্রেনিং ও টেস্ট ডেটার বিষয়টি বুঝে এলাম, সেইভাবে ডেটাসেটটিকে ভাগ করি, তাহলে ধরি, D1, D2, D3, D4 গুলি ট্রেনিং সেট-এ এবং D5 গুলি টেস্ট

অধ্যায় ২ : বিভিন্ন ধরনের লার্নিং ও অন্যান্য

সেট-এ। এটি আমরা র্যানডমলি সিঙ্কান্স নিলাম। আমরা শুধু D1, D2 ও D3-কেও ট্রেনিং সেট-এ দিয়ে D4 ও D5-কে টেস্ট সেটে দিতে পারতাম। এটি কীভাবে ভাগ হবে তার স্বাধীনতা পুরোপুরি প্রোগ্রামারের কাছে থাকবে।

এখন, আমরা যদি D1, D2, D3, D4 দিয়ে ট্রেইন করে D5 দিয়ে টেস্ট করে যে মডেল ইভ্যালুয়েশন পাব (মডেল ইভ্যালুয়েশন মানে হচ্ছে আমাদের মেশিন লার্নিং অ্যালগরিদম কতটুকু ভালো পারফরম করল, বা কতটুকু ভুলভাষ্টি করল সেটি বের করা), সেটাকে কি আমাদের পুরোপুরি সঠিক বলে ধরে নেওয়াটা ঠিক হবে? এটি অনেকটা ইশতিয়াকের প্রথম উদাহরণটির মতো হয়ে যাচ্ছে না, যেখানে এমন জিনিস পরীক্ষায় চলে এসেছে, যেটি সিলেবাসে ছিল না, যার ফলে ইশতিয়াক সেটি পড়েইনি? সাধারণত হ্যাঁ, আমরা টেস্ট ডেটা হিসেবে এমন ডেটা ব্যবহার করি, যেটি আমাদের মেশিন লার্নিং অ্যালগরিদম আগে কখনো দেখেনি। কিন্তু এই ক্ষেত্রে শুধু একটি অচেনা ডেটা সেট দিয়ে ইভ্যালুয়েশন করেই, আমাদের মডেল ভালো না খারাপ, সেই সিঙ্কান্সে উপনীত হওয়াটা ঠিক হবে না। আমাদের আরো একটু বেশি পরীক্ষা-নিরীক্ষা করতে হবে।

আবার, আমরা যদি, D1, D2, D3, D4, D5 সবগুলো ডেটা দিয়ে প্রথমেই একবারে ট্রেইন করে ফেলি এবং তারপর শুধু D1, D2, D3, D4 কিংবা D5 এদের যে-কোনো একটি দিয়ে টেস্ট করি, তাহলে সেটাও ভুল হবে। কেননা, আমাদের টেস্ট করতে হয় এমন ডেটা দিয়ে, যেটি আমাদের মেশিন লার্নিং অ্যালগরিদম আগে কখনো দেখেনি। কিন্তু এই ক্ষেত্রে মেশিন সমস্ত ডেটাই দেখে ফেলেছে।

আমাদের ট্রেনিং ও টেস্ট ডেটায় গোটা ডেটাসেট ভাগ করার সময় লক্ষ্য এটি থাকে যে আমরা ট্রেনিং ডেটা যত বেশি পারা যায় নেব, যাতে করে মেশিন ভালোভাবে শিখতে পারে; এবং টেস্ট ডেটা ও যত বেশি পারা যায় নেব, যাতে আমরা ভালোভাবে টেস্ট করতে পারি। আমরা যদি মাত্র 2-3টি ডেটা নিয়ে টেস্ট করে ভালো ফলাফল পেয়েই বলে দিই যে আমাদের মেশিন শিখে গেছে, তাহলে কি ব্যাপারটা ঠিক হবে? মোটেও নয়। আমাদের যত ভিম ভিম উপায়ে, যত ভালোভাবে পারা যায় টেস্ট করতে হবে, যাতে মেশিনের শেখায় কোনো ধরনের ফাঁকফোকর না থেকে যায়।

আর এখান থেকেই ক্রস ভ্যালিডেশনের উৎপত্তি। ক্রস ভ্যালিডেশনে যেটি হয় যে, আমরা প্রথমেই ডেটাসেটকে K-সংখ্যক সমান ভাগে ভাগ করে ফেলি। ভাগটা হয় র্যানডমলি। এটাকে K-fold Cross Validation ও বলা হয়। এরপর এই K-সংখ্যক ভাগ থেকে প্রতিবার (K-1)-সংখ্যক ভাগ দিয়ে মেশিন লার্নিং অ্যালগরিদমকে ট্রেইন করি এবং বাকি ভাগটি দিয়ে টেস্ট করি। তাই, যেটি হয় যে, আমাদের K-সংখ্যক ভাগের ভেতরে প্রতিটি ভাগই একবার-না-একবার টেস্ট হিসেবে ব্যবহৃত হবে। একটি লুপ চালিয়ে সেটাকে K-সংখ্যকবার ঘুরিয়ে এই কাজটি করা হয় এবং প্রতিটি ভিম ভিম টেস্ট সেটের জন্য ভিম ভিম পারফরম্যান্স ভ্যালু বের করা হয়। এরপর সবশেষে, সমস্ত

মেশিন লার্নিং আলগরিদম

পারফরম্যান্স ভ্যালুর গড়মান নেওয়া হয়, যেটি কিনা এই মডেল সত্ত্বাকারেই কতটুকু ভালো বা খারাপ, সেটি প্রকাশ করবে।

নিচের চার্ট (চার্ট 2.4.1) দেখুন :

Iteration (1 to K)	Training Set	Test Set	Performance Score
1	D2, D3, D4, D5	D1	S1
2	D1, D3, D4, D5	D2	S2
3	D1, D2, D4, D5	D3	S3
4	D1, D2, D3, D5	D4	S4
5	D1, D2, D3, D4	D5	S5

চার্ট 2.4.1

এখন তাহলে ফাইনাল মডেল ইভ্যালুয়েশন স্কোর হচ্ছে $= \frac{1}{k} \sum_{i=1}^k S_i$

অর্থাৎ, আমাদের এই উদাহরণের ফ্রেন্টে $= \frac{S1+S2+S3+S4+S5}{5}$

ক্রস ভ্যালিডেশন একটি চমৎকার জিনিস। এখানে একটু খেয়াল করলে দেখবেন, আমরা কৌশলে কিন্তু পুরো ডেটাসেটকেই ট্রেনিং ও টেস্ট উভয় কাজেই ব্যবহার করছি এবং এমনভাবে সেটি করছি, যাতে ট্রেনিং ও টেস্টের আসল ধারণাটুকুও বজায় থাকে। অর্থাৎ ব্যাপারটি কিন্তু আমাদের সেই ইশতিয়াকের উদাহরণটির মতো হয়ে গেল, যেখানে পুরো বই-ই সিলেবাসে ছিল এবং প্রশ্ন পুরো বই থেকেই হয়েছে। সুতরাং আমরা বলতে পারি যে, ক্রস ভ্যালিডেশন করে আমরা একটি মডেল ভালো না খারাপ – এ ব্যাপারে যে সিদ্ধান্তে পৌছাব সেটি মোটামুটি যুক্তিযুক্ত ও গ্রহণযোগ্য হবে। তো, মোটামুটি এই হচ্ছে ক্রস ভ্যালিডেশনের ধারণাটুকু। আশা করি, সবাই বুঝতে পেরেছেন।

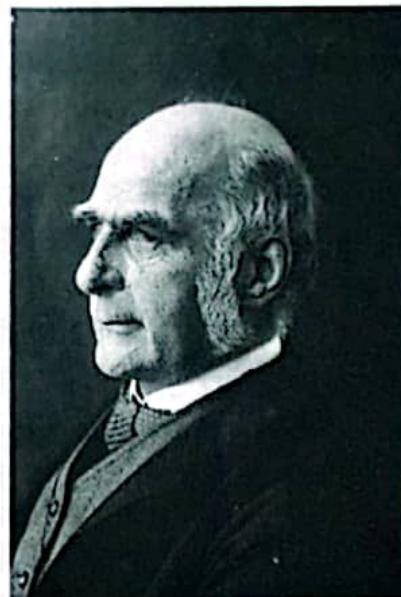
এরকম ছোটোখাটো আরো কিছু বিষয় আছে। সব এখানে একেবারে লিখলাম না। সামনে কাজ করার সঙ্গে সঙ্গে যখন যেটি দরকার হবে তখন সেটি লিখে দেব। আর আগেই বলেছি, এই বইতে মূলত কঠিন কঠিন থিওরি আর ম্যাথ আমরা একটু কম কম করে দেখিয়ে (যতটুকু না হলেই নয় ততটুকু শিখব) মেশিন লার্নিং আসলে কীভাবে ব্যবহার করা যায় সেটি উদাহরণ আকারে দেখব। আর চেষ্টা করব যতটা সহজ করে পারা যায়, বোঝাব।

তো শুরু করা যাক?

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

লিনিয়ার রিগ্রেশন শব্দের সঙ্গে অনেকে হয়তো পরিচিত, অনেকে নয়। যাঁরা পরিসংখ্যান বা স্ট্যাটিস্টিকস (Statistics) পড়েছেন, তাঁরা হয়তো জানেন এই পদ্ধতিটি সম্পর্কে। এটি আসলে পরিসংখ্যানের একটি পদ্ধতি। আর বর্তমানে এটি মেশিন লার্নিং অ্যালগরিদমগুলোর মধ্যে অন্যতম জনপ্রিয় একটি পদ্ধতি। এটি এত জনপ্রিয় হওয়ার মূল কারণ হয়তো এর সহজবোধ্যতা।

রিগ্রেশনের উভাবক বলা হয় স্যার ফ্রান্সিস গ্যালটনকে (Francis Galton, 1822 - 1911)। তিনিই প্রথম পরিসংখ্যানে এই রিগ্রেশনের ব্যবহার করেন।



ছবি 3.1 : Francis Galton (1822 – 1911)

পরিচ্ছেদ ৩.১ : লিনিয়ার রিগ্রেশন কী

তাত্ত্বিক আলোচনায় যাওয়ার আগে, আমাদের দৈনন্দিন জীবনের সঙ্গে মিলে যায়, এরকম একটি ঘটনা দিয়ে বোঝার চেষ্টা করা যাক লিনিয়ার রিগ্রেশন ব্যাপারটি কী। এটি আসলে আমার খুব প্রিয় একটি উদাহরণ। ক্লাসে আমার ছাত্রছাত্রীদের বোঝানোর সময়েও আমি এই উদাহরণটি ব্যবহার করি। এই উদাহরণটি আগেও একবার বর্ণনা করা হয়েছে। আপনাদের সুবিধার্থে আমি আবারও এটি বর্ণনা করছি এখানে।

ধরা যাক, আপনাকে বিভিন্ন সাইজের (ইঞ্জির) পিংজার দাম দেওয়া আছে (আগের অধ্যায়ে দেওয়া উদাহরণের মতো)। এই ডেটার ওপরে ভিত্তি করে আপনাকে এখন কাজ করতে হবে। আপনার কাছে থাকা ডেটা টেবিল 3.1.1-এ দেওয়া হলো।

এখন এই দামগুলোর ওপর ভিত্তি করে, আপনাকে যদি আন্দাজ করতে দেওয়া হয় যে 17 ইঞ্জিং কোনো একটি পিংজা যদি বানানো হয়, তবে সেই পিংজার দাম কত হবে, তাহলে কি আপনি সেটি আন্দাজ করতে পারবেন? যদি পারেন তাহলে সেটি কীভাবে করবেন?

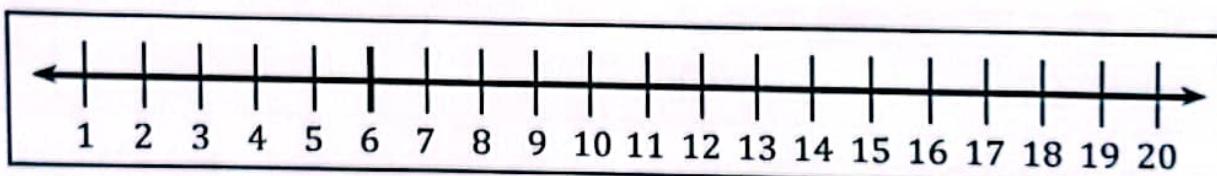
মেশিন লার্নিং অ্যালগরিদম

পিংজার সাইজ (ইঞ্চিতে)	পিংজার দাম (টাকায়)
6"	350/-
8"	775/-
12"	1150/-
14"	1395/-
18"	1675/-

টেবিল 3.1.1

আপনি প্রথমেই দেখবেন যে 17 সংখ্যাটির বৈশিষ্ট্য কী। এটি 14-এর চেয়ে বড়ো, কিন্তু 18-এর চেয়ে ছোটো, তাই না? তার মানে 17 ইঞ্চি পিংজার দাম অবশ্যই 14 ইঞ্চি পিংজার দামের চেয়ে বেশি হবে, কিন্তু 18 ইঞ্চি পিংজার দামের চেয়ে কম হবে।

আবার দেখুন, আমরা যদি একটি সংখ্যারেখার কথা চিন্তা করি (ছবি 3.1.1), তাহলে আমরা খুব সহজেই এটি বলতে পারি যে 17 সংখ্যাটি 14 ও 18-এর মধ্যে 18-এর বেশি কাছাকাছি, তাই না? এই তথ্যের ওপর ভিত্তি করে আমরা এও বলতে পারি, এখন যে 17 ইঞ্চি পিংজার দাম 18 ইঞ্চি পিংজার দামের কাছাকাছি কিছু একটা হবে।



ছবি 3.1.1

একটু ভালো করে চিন্তা করে দেখুন, আপনাকে কেউ যদি 17 ইঞ্চি পিংজার দাম আন্দাজ করতে দিত, আপনি মনে মনে ঠিক এভাবেই চিন্তা করতেন। শুধু আপনি অনেক দ্রুত এবং নিজের অজান্তেই এটি চিন্তা করে ফেলেন দেখে ঠিক অনেক সময় বুঝাতে পারেন না যে, আসলে চিন্তাটি কীভাবে করলেন।

আপনি যদি ওপরের ঘটনাটি বুঝে থাকেন, তাহলে আপনি লিনিয়ার রিগ্রেশন বুঝে গেছেন অনেকখানিই।

তাহলে লিনিয়ার রিগ্রেশন বিষয়টি আসলে কী? গণিতের ভাষায় বললে এটি অধীন চলক (Dependent Variable) ও স্বাধীন চলকের (Independent Variable) মধ্যেকার সম্পর্ক নির্ণয়ের একটি মাধ্যম!

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

একেবারেই দুর্বোধ্য মনে হচ্ছে, তাই না? আচ্ছা চলুন দেখি, এখন গণিতের ভাষার সঙ্গে আমাদের পিংজার ব্যাপারটি মেলানো যায় কি না।

আপনারা যদি একটু ভালো করে পিংজার সাইজ ও দামগুলো লক্ষ করেন, তাহলে দেখবেন, যে পিংজার সাইজ যত বেড়েছে, তার দাম তত বেড়েছে। তার মানে, পিংজার দাম কত হবে সেটি নির্ভর করছে তার সাইজের ওপর। সাইজ যত বড়ো, দাম তত বেশি। তাহলে, এখানে পিংজার দামটা পিংজার সাইজের ওপর নির্ভরশীল, তাই না?

তাহলে কি আমরা বলতে পারি না যে, এখানে পিংজার সাইজ হলো আমাদের স্বাধীন চলক (Independent Variable) এবং পিংজার দাম হলো অধীন চলক (Dependent Variable)?

আচ্ছা, এই IV (Independent Variable)-কে এখন থেকে IV লিখব, আর Dependent Variable-কে লিখব DV) আর DV-এর মধ্যে কী সম্পর্ক সেটি নিয়ে একটু কথা বলি! এখানে IV আর DV-এর মধ্যে সম্পর্ক বলতে আসলে বোঝাবে IV-এর দাম এক একক (1 unit) বৃদ্ধি পেলে DV-এর দাম কতটুকু বৃদ্ধি পাবে, সেটি নির্ণয়ের চেষ্টা করা! এটিই লিনিয়ার রিগ্রেশন।

এ তো বললাম কঠিন গণিতের ভাষার, এখন খাবারের ভাষায় বলি। ওপরের উদাহরণ থেকে দেখুন, আমাদের পিংজার সাইজ প্রতিবার বিভিন্ন আকারে বৃদ্ধি পেয়েছে। প্রথমে ছিল 6 ইঞ্চি, এর পরে 8 ইঞ্চি, 12 ইঞ্চি, 14 ইঞ্চি, 18 ইঞ্চি ইত্যাদি। আমাদের এখান থেকে একটু বের করার চেষ্টা করতে হবে যে পিংজার সাইজ 1 ইঞ্চি বৃদ্ধি পেলে আসলে দাম গড়ে কতটা বাড়ে। এই হিসাবটি আমরা যত সূক্ষ্মভাবে বের করতে পারব, তত আমাদের হিসাব নির্ভুল হবে।

তাহলে মোদ্দা কথাটা কী দাঁড়াল? স্বাধীন চলকের মানের পরিবর্তনের সঙ্গে সঙ্গে কীভাবে অধীন চলকের মানের পরিবর্তন হয়, অর্থাৎ স্বাধীন ও অধীন চলকের মধ্যেকার সম্পর্কটি আসলে কী, সেটি নির্ণয়ের একটি পদ্ধতিই হলো লিনিয়ার রিগ্রেশন! এবং এই সম্পর্কটি যদি আমরা একবার বের করে ফেলতে পারি আমাদের জানা অধীন ও স্বাধীন চলকের মান থেকে, তাহলে সেই সম্পর্ক ব্যবহার করে আমরা স্বাধীন চলকের যে-কোনো অজানা মানের জন্য অধীন চলকের মান বের করে ফেলতে পারি।

অর্থাৎ, এই যে 5টি ভিন্ন ভিন্ন সাইজের পিংজার দাম ও সাইজ যে আমাদের দেওয়া ছিল, সেখান থেকে আমরা যদি সাইজ ও দামের মধ্যে একটি সহজ-সরল সম্পর্ক বের করে ফেলতে পারি, তাহলে সেই সম্পর্ক ব্যবহার করে খুব সহজেই এমন কোনো সাইজের পিংজা, যার দাম আমরা জানি না (যেমন 17 ইঞ্চি পিংজা), তার দামও আন্দাজ করে বলে দিতে পারি। দামটি পুরোপুরি সঠিক না হলেও, খুব কাছাকাছি একটি মান হবে। আমাদের এখনকার কাজের জন্য সেটুকুই যথেষ্ট।

পরিচ্ছন্দ ৩.২ : নিউমেরিকাল অ্যানালাইসিস (Numerical Analysis) ব্যবহার করে লিনিয়ার রিগ্রেশন

লিনিয়ার রিগ্রেশন দুই পদ্ধতিতে করা যায়। প্রথম পদ্ধতিটি অপেক্ষাকৃত সহজ, শুধু সূত্র প্রয়োগ করতে হবে, আর কিছু নয়। এটি একটি নিউমেরিকাল অ্যানালাইসিস মেথড। তাই এটি দিয়েই শুরু করি।

আমাদের টেবিল 3.1-এ দেওয়া ডেটাসেটটি নিয়েই আমরা কাজ করব এবং এটি দিয়েই আমাদের ধারণাটি বুবাব। দেখি কতদূর এগোতে পারি। ব্যবহারের সুবিধার জন্য আমাদের ডেটাসেটটি (পিংজার দাম ও সাইজের টেবিলটিকে আমরা ডেটাসেট বলব) আবার এখানে দিচ্ছি (টেবিল 3.2.1)।

পিংজার সাইজ (ইঞ্চিতে)	পিংজার দাম (টাকায়)
6"	350/-
8"	775/-
12"	1150/-
14"	1395/-
18"	1675/-

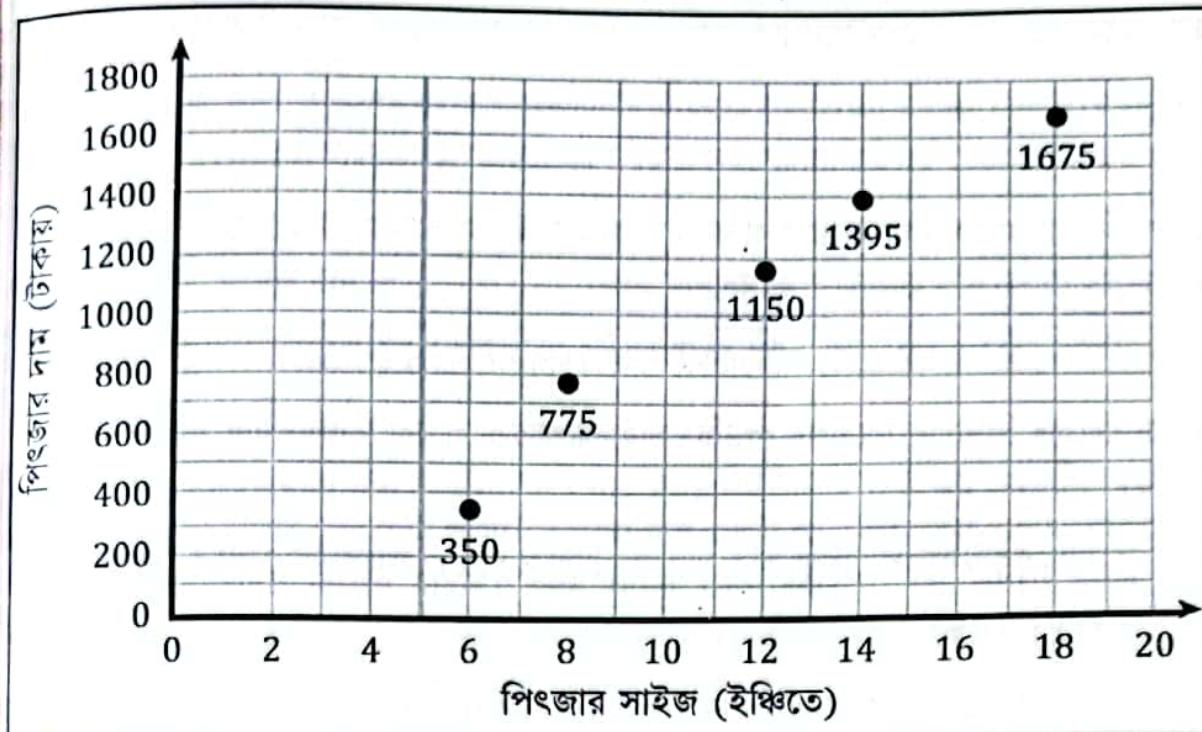
টেবিল : 3.2.1

আমরা বলেছিলাম, আমাদের লক্ষ্য হচ্ছে, যে ডেটাসেট আমাদের কাছে আছে, সেটি ব্যবহার করে আমাদের বের করতে হবে 17 ইঞ্চি পিংজার দাম কত টাকা, তাই না?

দাম বের করার আগে আমরা যদি একটি গ্রাফ আঁকার চেষ্টা করি, যেখানে X-অক্ষ বরাবর নেব আমাদের স্বাধীন চলক, যেটি হলো পিংজার সাইজ এবং Y-অক্ষ বরাবর নেব আমাদের অধীন চলক, যা হলো পিংজার দাম। তাহলে আমাদের গ্রাফের চেহারাটি দাঁড়াবে ছবি 3.2.1-এর গ্রাফের মতো।

আশা করছি, গ্রাফটি সবাই বুঝতে পেরেছেন? এই গ্রাফে দেখুন 5টি বিন্দু বা পয়েন্ট আছে, আমরা এগুলোকে বলব আমাদের ডেটা পয়েন্ট। এখন আসি মূল কথায়। আমাদের যেটি করতে হবে, এই ডেটাপয়েন্টগুলোর মধ্যে আমাদের একটি সম্পর্ক বের করে ফেলতে হবে, তাই না? যদি ঠিক বের করতে পারি, তাহলেই কেবল সেই সম্পর্কটি ব্যবহার করে আমরা নতুন অজানা সাইজের পিংজার দাম বের করে ফেলতে পারব।

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)



গ্রাফ 3.2.1: পিংজার সাইজ ও দামের মধ্যে সম্পর্ক

এখন, কতগুলো বিন্দুর মধ্যে সবচেয়ে সহজ-সরল সম্পর্ক কী হতে পারে? উভয় হচ্ছে – একটি সরলরেখা। আমরা যদি এমন একটি সরলরেখা বের করতে পারি, যার ওপরে আমাদের সবগুলো ডেটা পয়েন্ট অবস্থান করে, তাহলে কী হবে? ওই সরলরেখার সমীকরণে প্রতিটি বিন্দুর মান বসালে সমীকরণের দুই পক্ষ সমান পাব। মনে করে দেখুন, ছোটোবেলায় আমরা পড়েছিলাম, যে-কোনো সরলরেখার সমীকরণে ওই সরলরেখার ওপরের কোনো বিন্দুর মান বসালে আমরা সমীকরণটিকে সিদ্ধ করতে পারি, এরকম একটি কথা ছিল।

তাই, আমাদেরকে একটি সরলরেখা আঁকার চেষ্টা করতে হবে। লক্ষ্য থাকবে এমন একটি সরলরেখা খুঁজে বের করা, যা আমাদের সবগুলো ডেটা পয়েন্টের ওপর দিয়ে যায়, অর্থাৎ সবগুলো ডেটা পয়েন্ট আমাদের সরলরেখাটির সমীকরণকে সিদ্ধ করে। সেরকম যদি কোনো সরলরেখা না-ও পাই, অন্তত এমন কোনো সরলরেখা বের করতে হবে, যা থেকে আমাদের সবগুলো ডেটাপয়েন্টের দূরত্ব সর্বনিম্ন হয়।

এখানে লক্ষণীয়, আমরা হয়তো-বা সরলরেখা না এঁকে আঁকাবাঁকা বক্ররেখা বা কার্ভ (Curve) আঁকার চেষ্টা করতে পারি। কেননা এমনও হতে পারে যে, আমরা একটি সরলরেখা আঁকলে সেটি সবগুলো ডেটা পয়েন্ট দিয়ে যাচ্ছে না কিন্তু যদি একটি কার্ভ আঁকি, তাহলে সেটি সবগুলো ডেটা পয়েন্ট দিয়ে যাচ্ছে। কিন্তু, সে ক্ষেত্রে তাহলে সেটি আর লিনিয়ার বা সরলরেখিক থাকবে না (সমীকরণের হিসেবে যদি বলি, x -এর সর্বোচ্চ ঘাত আর ১ থাকবে না, এর বেশি হয়ে যাবে)।

ଏକଟୁ ମନେ କରେ ଦେଖୁନ, ହୋଟୋବେଲାଯ ଆମାଦେର ପଡ଼ା ସରଳରେଖାର ସମୀକରଣ ଛିଲ,

$$Y = mX + c \quad (\text{ଏଥାନେ } X \text{ ହଜ୍ଜେ ସ୍ଵାଧୀନ ଚଲକ, } Y \text{ ହଜ୍ଜେ } X\text{-ଏର ଫାଂଶନ ଏବଂ ସମୀକରଣଟି ଲିନିୟାର, କେନନା ସ୍ଵାଧୀନ ଚଲକର ସର୍ବୋତ୍ତମାନ } 1)$$

ଏଥନ ଯଦି ଆମରା କୋନୋ ବକ୍ରରେଖା ବା କାର୍ଡ ଆଂକତେ ଯାଇ, ତଥନ ଅବଶ୍ୟକ ଆମାଦେର X -ଏର ଘାତେ ମାନ 1-ଏର ଚେଯେ ବେଶି ହତେ ହବେ, ଯେମନ,

$$Y = aX^2 + bX + C, \text{ ଏହି ଏକଟି ବକ୍ରରେଖାର ସମୀକରଣ (ପ୍ଯାରାବୋଲା ବା ଅଧିବୃତ୍ତ)}$$

ଆମରା ଯେହେତୁ ଲିନିୟାର ରିଗ୍ରେଶନ ପଡ଼ିଛି, ତାଇ ଆପାତତ ଆମରା ସରଳରେଖାର ମଧ୍ୟେ ଦୀର୍ଘବନ୍ଧ ଥାକଛି।

ଏଥନ, ଆମରା ଏକଟି ଫାଂଶନ ଚିନ୍ତା କରି, ଯାର ନାମ ଦିଲାମ $f(x)$ । ଫାଂଶନଟି ଦେଖତେ ହବେ ଏକମ -

$$f(X) = Y$$

ଅର୍ଥାତ୍, ଫାଂଶନଟିତେ ଆମରା X -ଏର ମାନ (ପିଂଜାର ସାଇଜ) ଇନପୁଟ ଦିଲେ, ଫାଂଶନଟି ଆମାଦେର ଆଉଟପୁଟ ଦେବେ Y , ଯେତି କିନା ପିଂଜାର ଦାମ । ଆମରା ତୋ ଏହି ଜାଦୁର ଫାଂଶନଟିଇ ଖୁଜେ ବେର କରତେ ଚାଇଛି, ତାଇ ନୟ କି? ମନେ କରେ ଦେଖୁନ, ଆମରା ପଡ଼େଛିଲାମ, ଫାଂଶନ ହଜ୍ଜେ ଏକ ଧରନେର ଅନ୍ଧର ବା ସମ୍ପର୍କ, ଆମାଦେର ଇନପୁଟ ଓ ଆଉଟପୁଟେର ଭେତରେ । ଆମରା ଏହି ସମ୍ପର୍କଟିଇ ଖୁଜେ ବେର କରତେ ଚାଇଛି ।

ଆମରା, $f(X) = mX + c$ ଲିଖତେ ପାରି ଅବଶ୍ୟକ, ଯେହେତୁ ଆମରା ସରଳରେଖା ବେର କରତେ ଚାଇଛି, ତାଇ ଏଟାଇ ହବେ ଆମାଦେର ଫାଂଶନେର ଗାଣିତିକ ରୂପ । ତାହଲେ, ଶେଷମେଶ ଦାଁଡ଼ାଳ -

$$Y = mX + c$$

ଏଥନ, ଆମାଦେର ଏହି ସମୀକରଣେ m ହଲୋ ଢାଳ ବା ସ୍ଲୋପ, ଯେତି ସରଳରେଖାର ଦିକ ନିର୍ଧାରଣ କରେ । ମାନେ, ସେଠି X -ଅକ୍ଷେର ଧନାତ୍ମକ ଦିକେର ସଙ୍ଗେ କତ ଡିଗ୍ରି କୋଣ କରେ ଆଛେ ତାର Tangent-ଏର ମାନକେ ବୋଲାଯ । ଆର c ହଲୋ Y -ଅକ୍ଷେର କର୍ତ୍ତିତ ଅଂଶ ବା Intercept (Intercept)-ଏର ମାନ, ଅର୍ଥାତ୍ ସରଳରେଖାଟିଇ Y -ଅକ୍ଷକେ କୋଣ ବିନ୍ଦୁତେ ଛେଦ କରେଛେ ତାର ମାନ ।

ଏଥନ, m ଓ c -ଏର ଭିନ୍ନ ଭିନ୍ନ ଅସୀମସଂଖ୍ୟକ ମାନେର ଜନ୍ୟ ଆମରା କିନ୍ତୁ ଅନ୍ୟ ସରଳରେଖା ପାର । ସେଇ ସରଳରେଖାଗୁଲୋ X -ଅକ୍ଷେର ଧନାତ୍ମକ ଦିକେର ସଙ୍ଗେ ବିଭିନ୍ନ କୋଣ କରେ ଥାକବେ ଏବଂ ତାରା Y -ଅକ୍ଷକେ ବିଭିନ୍ନ ବିନ୍ଦୁତେ ଛେଦ କରବେ ।

ଆମରା ଧରେ ନିଇ ଯେ, M_{final} ଓ C_{final} ହଜ୍ଜେ m ଓ $21c$ -ଏର ସେଇ ଦୁଟି ବିଶେଷ ମାନ, ଯାର ଜନ୍ୟ ଆମରା ଆମାଦେର କାଞ୍ଚିତ ସରଳରେଖାଟି ପେଯେ ଯାବ ଯେତି ଆମାଦେର ସମସ୍ତ ଡେଟା ପଯେନ୍ଟେର ଓପର ଦିଯେ ଯାବେ । ତାହଲେ, ମାନଗୁଲୋ ସମୀକରଣେ ବସିଯେ ପାଇ -

অধ্যায় ৩ : সিনিয়ার রিফ্রেশন (Linear Regression)

$$Y = (X \times M_{final}) + C_{final}$$

এই ফাংশনটি বের করাই হচ্ছে আমাদের লক্ষ্য বা টার্গেট, তাই আমরা একে বলব টার্গেট (Target) ফাংশন বা ট্রু (True) ফাংশন। এই ফাংশনের সরলরেখাটি আমাদের সব ডেটা পয়েন্টের ওপর দিয়ে যাবে।

কিন্তু কথা হচ্ছে, আমরা তো M_{final} ও C_{final} -এর মান জানি না। তাই, আমরা যানডমলি দৃষ্টি মান দিয়ে তুক করব। ধরি সেগুলো m_1 ও c_1 , সূতৰাং সরলরেখাটির সমীকরণ তখন হবে,

$$Y = X \times m_1 + c_1$$

হ্যাতো আমরা শুরুতে দেখলাম, আমাদের এই m_1 ও c_1 -এর মানের জন্য এমন একটি সরলরেখা পেয়েছি, যেটি আমাদের ৫টি ডেটা পয়েন্টের মধ্যে মাত্র একটি ডেটা পয়েন্টের ওপর দিয়ে গোছে, বাকি ৪টির ধারেকাছে দিয়েও যায়নি। তখন আমরা যোগ, বিয়োগ, গুণ, ভাগ ইত্যাদি নানা গণিতিক অপারেশন চালিয়ে ঢেঁক করব আমাদের m_1 ও c_1 -এর মানগুলোকে M_{final} ও C_{final} -এর মানের যত কাছাকাছি পারা যায় নিয়ে আসতে, যাতে করে আমাদের সরলরেখাটি সেগুলো ডেটা পয়েন্টের ওপর দিয়ে যায়।

এই যে, প্রতিবার আমরা একেকটি m ও c -এর মান পাওছি (যেমন m_1 , c_1 পেলাম ইত্যাদি) সেগুলোকে আমাদের ফাংশনে বসালে আমরা যে সমীকরণটি পাই, তাকে বলা হয় হাইপোথিসিস। সমস্ত সম্ভাব্য হাইপোথিসিস নিয়ে তৈরি হয় হাইপোথিসিস স্পেস (Hypothesis Space)। তার মাধ্য থেকে যেটি সবচেয়ে ভালো ফলাফল দেয়, আমরা তাকেই আমাদের ফোইনাল হাইপোথিসিস হিসেবে বেছে নিই।

যেমন, $Y = X \times m_1 + c_1$, এটি একটি হাইপোথিসিস। এই হাইপোথিসিস হচ্ছে এমন একটি ফাংশন, যেটিকে আমরা আমাদের টার্গেট ফাংশনের সঙ্গে মিলে যেতে পারে বলে ধরে নিই এবং পর হিসাবনিকাশ করে দেখ যে আসলেই মিলে যাচ্ছে কি না।

আমরা m_1 ও c_1 -এর ওপরে গাণিতিক বিভিন্ন অপারেশন চালিয়ে, ধরি, নতুন আরো দৃষ্টি মান পেলাম - m_2 ও c_2 । সূতৰাং তখন আমাদের বিবেচনার জন্য নতুন হাইপোথিসিস হবে

$$Y = X \times m_2 + c_2$$

এবং আমরা আবার মিলিয়ে দেখব সেটি আমাদের টার্গেট ফাংশনের মতো আচরণ করে কি না, অর্থাৎ সবগুলো ডেটাপয়েন্টের ওপর দিয়ে যায় কি না, বা কাছাকাছি যায় কি না।

এই হাইপোথিসিসের ধারণাটি মেশিন লার্নিংয়ে অত্যন্ত গুরুত্বপূর্ণ। তাই আমি বলব, কেউ যদি এখনো এই ধারণাটি পরিকারভাবে না বুঝে থাকেন, তাহলে অনুচ্ছেদের শুরু থেকে আরেকবার পড়ুন।

মেশিন লার্নিং আলগরিদম

এখন, আমরা ইতিমধ্যেই বলেছি, এত এত অসংখ্য সরলরেখার ভিত্তে আমাদের এমন একটি সরলরেখা খুঁজে বের করতে হবে, যে সরলরেখা থেকে এই 5টি বিন্দুর দূরত্ব অন্য যে-কোনো সরলরেখার চেয়ে কম হয়। সেই সরলরেখাটিকে আমরা অনেক সময় বলে থাকি রিগ্রেশন (Regression) লাইন, এটিই আমাদের কাঞ্জিক্ত সমাধান।

এই সরলরেখাটির আরো একটি সুন্দর নাম আছে, এটাকে বলা হবে বেস্ট ফিট লাইন (Best Fit Line)। এর কারণ হচ্ছে সেই সরলরেখাটা আমাদের ডেটার ভেতরে সবচেয়ে ভালো ফিট করবে কিংবা সবচেয়ে ভালো সমাধান দেবে।

যা-ই হোক, আমরা আমাদের সরলরেখার অনুসন্ধানে ফেরত আসি। সরলরেখার সমীকরণের এই m ও c -এর মান ডেটাসেট থেকে সরাসরি পাওয়ার জন্য দুটি সূত্র আছে, যেটি নিউমেরিকাল অ্যানালাইসিস পদ্ধতি প্রয়োগ করে বের করা। সূত্র দুটি নিম্নরূপ –

$$m = \frac{\bar{x} \cdot \bar{y} - \bar{xy}}{(\bar{x})^2 - \bar{x}^2}$$

$$c = \bar{y} - m\bar{x}$$

এখন এই সূত্র দুটি টেবিল 3.2.2-এর ডেটার ওপরে ব্যবহার করে আমরা m ও c -এর মান নির্ণয় করে ফেলি চলুন :

X	Y	XY	X^2
6	350	2100	36
8	775	6200	64
12	1150	13800	144
14	1395	19530	196
18	1675	30150	324

টেবিল 3.2.2

এখন m -এর মান বের করার জন্য যা যা লাগবে, চলুন সেগুলো বের করে ফেলি –

$$\bar{X} = \frac{6 + 8 + 12 + 14 + 18}{5} = 11.6$$

$$\bar{Y} = \frac{350 + 775 + 1150 + 1395 + 1675}{5} = 1069$$

$$\bar{XY} = \frac{2100 + 6200 + 13800 + 19530 + 30150}{5} = 14356$$

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

$$(\bar{X})^2 = (11.6)^2 = 134.56$$

$$\bar{x^2} = \frac{36 + 64 + 144 + 196 + 324}{5} = 152.8$$

সূতরাং

$$m = \frac{(11.6 \times 1069) - 14356}{134.56 - 152.8} = \frac{-1955.6}{-18.24} = 107.21$$

এখন m , \bar{y} ও \bar{x} মান (3) নম্বরে বসিয়ে পাই,

$$C = 1069 - (107.21 \times 11.6) = -174.6$$

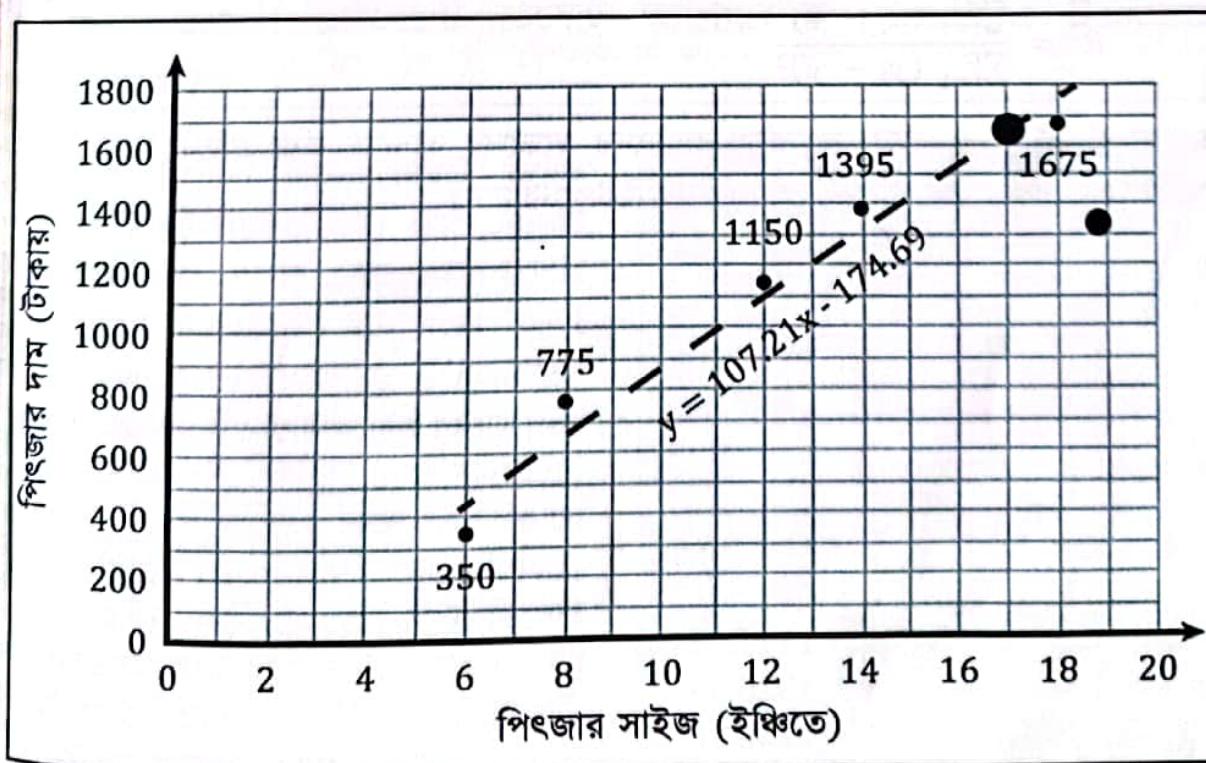
সূতরাং আমাদের কাঞ্চিত সরলরেখার সমীকরণ দাঁড়াল –

$$y = 107.21x - 174.6$$

এটিই আমাদের রিগ্রেশন লাইন বা বেস্ট ফিট লাইন।

এখন এই রেখার সমীকরণে যদি আমরা $x = 17$ বসাই, তাহলেই আমরা পেয়ে যাব আমাদের 17 ইঞ্জিং পিংজার দাম,

যা আসলে হবে $= (107.21 \times 17) - 174.6 = 1647.96$ টাকা। \checkmark



গ্রাফ 3.2.2 : পিংজার সাইজ ও দামের মধ্যে সম্পর্ক

এবং আমরা যদি এখন আমাদের প্রাফ 3.2.1-এ আমাদের রিপ্রেশন লাইন এবং নতুন পাওয়া 17 ইঞ্জিনিয়ার দাম যোগ করি তাহলে আমরা প্রাফ 3.2.2-এর মতো একটি প্রাফ পাব।

প্রাফটিতে আমরা ধূসর রঙের ডটেড মেই লাইনটি দেখতে পাচ্ছি, সেটিই আমাদের রিপ্রেশন লাইন। আর এই লাইনের ওপরের কালো রঙের একটি বড়ো বিন্দু দেখতে পাচ্ছি আমরা, সেটিই হলো আমাদের নতুন পাওয়া ডেটা পয়েন্ট – (17, 1647.96)।

এখানে আরেকটি বিষয় আপনাদের জানিয়ে রাখি। আপনার লিনিয়ার রিপ্রেশনের হাইপোথিসিস আসলে কতটুকু ভালো পারফরম করছে, অর্থাৎ আপনার হাইপোথিসিস কত ভালো ফিট করছে আপনার ডেটার ভেতরে, সেটি পরীক্ষা করার জন্য লিনিয়ার রিপ্রেশনের ক্ষেত্রে একটি মান বের করা হয়, যেটাকে বলা হয় আর-স্ক্যারড মান (R-Squared Value)। এর মান 0% থেকে 100%-এর মধ্যে যে-কোনো কিছু হতে পারে। এই R-Squared মান হচ্ছে একটি ভালো-মন্দের সূচকের মতো। কোনো হাইপোথিসিসের জন্য মান যত বেশি হবে, আমরা বুঝে নেব সেই হাইপোথিসিস মডেল তত ভালোভাবে ফিট করবে ডেটার ভেতর। এটি নিয়ে বিস্তারিত আলোচনা করা হয়েছে পরিচ্ছেদ ১৩.১-এ।

আমাদের এই সরলরেখার ক্ষেত্রে R-Squared Value-এর মান বের করার জন্য আমরা নিচের সূত্রটি ব্যবহার করব –

$$R^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

এখানে \hat{y}_i , \bar{y} ও y_i হচ্ছে যথোক্তমে আমাদের মডেলের আন্দাজ করা প্রতিটি মান, সমস্ত আউটপুটের প্রকৃত মান এবং প্রতিটি প্রকৃত আউটপুটের মান।

আমাদের কাঙ্ক্ষিত সরলরেখার সমীকরণ ছিল –

$$y = 107.21x - 174.6$$

এখন আমরা এই সমীকরণ ব্যবহার করে x -এর বিভিন্ন মানের জন্য আমাদের লিনিয়ার মডেলের প্রেডিকশন বের করে ফেলতে পারি।

$X = 6,$	$y = 468.66$
$X = 8,$	$y = 683.08$
$X = 12,$	$y = 1111.92$
$X = 14,$	$y = 1326.34$
$X = 18,$	$y = 1755.18$

মানগুলোকে R-Squared Value-এর সূত্রে বসিয়ে পাই,

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

$$R^2 = \frac{(468.88 - 1069)^2 + (683.08 - 1069)^2 + (1111.92 - 1069)^2 + (1326.34 - 1069)^2 + (1755.18 - 1069)^2}{(350 - 1069)^2 + (775 - 1069)^2 + (1150 - 1069)^2 + (1395 - 1069)^2 + (1675 - 1069)^2}$$
$$= \frac{360144.01 + 148934.26 + 1842.13 + 66223.88 + 470842.99}{516961 + 86436 + 6561 + 106276 + 367236}$$
$$= 0.96$$

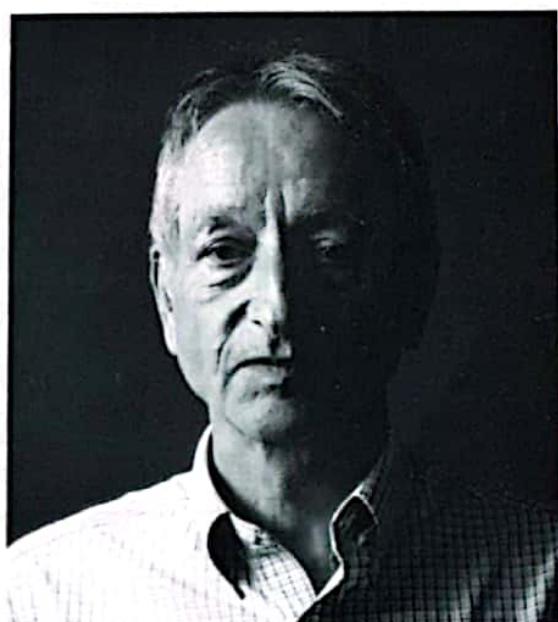
সব সময় আমাদের লক্ষ্য থাকবে, R-Squared মান ম্যাক্সিমাইজ (Maximize) করে এমন হাইপোথিসিস খুঁজে বের করা। অর্থাৎ m ও c -এর এমন মান বের করা যাব জন্য R-Squared মান 100% বা এর সর্বোচ্চ যতটুকু কাছাকাছি যাওয়া সম্ভব, ততটুকু যায় এবং আমরা নিউমেরিকাল অ্যানালাইসিস মেথড প্রয়োগ করে সরাসরি কীভাবে সেই m ও c -এর মান বের করা যায়, সেটি একটু আগেই দেখেছি।

আপনারা যদি এ পর্যন্ত মোটামুটি বুঝে থাকেন, তাহলে আপনারা এখন নিউমেরিকাল অ্যানালাইসিস মেথড প্রয়োগ করে লিনিয়ার রিগ্রেশন অনেকখানিই বুঝে গেছেন এবং এখন আপনারা এটি নিজেরাই হাতে কলমে কোড করতে পারবেন বলে আশা করছি।

তবে আর দেরি কেন, এখনই বসে পড়ুন কোড করতে।

পরিচ্ছেদ ৩.৩ : গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম (Gradient Descent Algorithm)

গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদমটি অনেক পুরোনো একটি অ্যালগরিদম। এর সূচনা হয় অগাস্টিন লুইস কসির (Augustin-Louis Cauchy; 1789-1857) হাত ধরে। তিনিই প্রথম *Comptes rendus de l'Académie des Sciences* জার্নালে এই গ্রেডিয়েন্ট ডিসেন্ট নামে নন লিনিয়ার অপটিমাইজেশন মেথডটি উন্নোত্ত করেন। কিন্তু পরবর্তীকালে জেফরি হিন্টন (Geoffrey Hinton, 1947) প্রথম মেশিন লার্নিংয়ে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ব্যবহার করেন। সুতরাং জেফরি হিন্টনকেই মেশিন লার্নিংয়ে গ্রেডিয়েন্ট ডিসেন্ট-এর জনক বলা চলে।



ছবি 3.3.1 : জেফরি হিন্টন (Geoffrey Hinton, 1947)

গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ব্যবহার করে লিনিয়ার রিপ্রেশন করার আগে আমাদের একটু বুঝে নিতে হবে যে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম আসলে কী এবং কীভাবে কাজ করে।

একটু আগে আমরা কতগুলো সূত্রে মান বসিয়ে সরাসরি হিসাব কয়ে m ও c -এর মান বের করেছি, যা মোটামুটি সঠিক। এখন চলুন, আমরা এই সূত্র দিয়ে মান বের না করে অন্য একটি পদ্ধতিতে চিন্তা করি। ধরা যাক, আমাদের যেহেতু m ও c -এর মান জানা নেই, তাই আমরা প্রথমেই আন্দাজ (randomly) আমাদের m ও c -এর দুটো মান ধরে নিই। এটি আপনারা যে-কোনো কিছু ধরতে পারেন। আমি ধরে নিলাম $m = 50, c = 100$ । আপনারা অন্য কোনো মান ধরতে পারেন, তাতে কোনো অসুবিধা নেই।

তাহলে, এখন আমাদের সরলরেখার সমীকরণ দাঁড়াল $y = 50x + 100$ এবং এটিতে আমাদের জানা x -এর মানগুলো থেকে যে-কোনো একটি যদি বসাই (যেমন, $x = 6$), তাহলে আমরা পাই $y = 400$ । এটি কিন্তু আমাদের আসল দামের সঙ্গে মিল না, কেননা আসল দাম দেওয়া আছে 350 টাকা। এখানে যেটি হলো, আমরা আসলে শুরুতে ভুল আন্দাজ করেছিলাম m ও c -এর মান, যার কারণে আমরা $400 - 350 = 50$ টাকার মতো গরমিল পাচ্ছি।

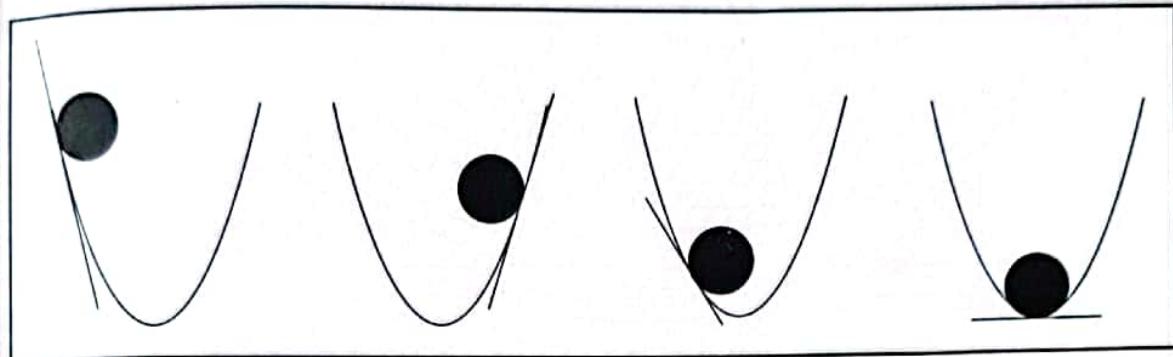
এজন্য আমাদের এখন যে কাজটি করতে হবে তা হলো, m ও c -এর মান পরিবর্তন বা আপডেট করতে হবে যাতে আমাদের এই ভুল (error = 50 টাকা গরমিল)-এর পরিমাণ কমে আসে। আমাদের লক্ষ্য হলো ভুলের পরিমাণ শূন্য বা এর কাছাকাছি নামিয়ে নিয়ে আসা। অর্থাৎ m ও c -এর মান আপডেট করতে করতে আমাদের ততক্ষণ পর্যন্ত আপডেট করতে হবে যতক্ষণ পর্যন্ত না আমাদের জানা প্রতিটি ট্রেনিং ডেটার জন্য এই ভুলের পরিমাণ শূন্য বা শূন্যের যত কাছাকাছি নামিয়ে আনা যায়, তত কাছাকাছি নেমে আসে। এই প্রতিবার মান আপডেট করে করে ভুল (বা Cost বা Error)-এর মান নামিয়ে নিয়ে আসার পদ্ধতিই হচ্ছে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম।

একে গ্রেডিয়েন্ট ডিসেন্ট (Gradient Descent) বলা একটি কারণ আছে। গ্রেডিয়েন্ট হলো সোজা করে বলতে গেলে ঢাল। এখন প্রশ্ন হচ্ছে কীসের ঢাল? কোনো একটি গ্রাফে যদি কোনো একটি ফাংশন প্লট করা হয়, তাহলে সেই গ্রাফের যে-কোনো একটি বিন্দুতে একটি স্পর্শক আঁকলে, সেটি X অক্ষের সঙ্গে যে কোণ উৎপন্ন করে থাকে তাঁর ট্যানজেন্ট (Tangent)-এর মানকে বলে গ্রেডিয়েন্ট। আর ডিসেন্ট শব্দের মানে হলো কমানো বা কমে যাওয়া। তাঁর মানে গ্রেডিয়েন্ট ডিসেন্ট-এর মানে অনেকটা এরকম দাঁড়ায় যে ঢালের মান কমানো এবং কমাতে কমাতে শূন্যের কাছাকাছি নিয়ে আসা। শূন্যতে নিয়ে যেতে পারলে আরো ভালো।

গ্রেডিয়েন্ট ডিসেন্ট আসলে কী, সেটা বুঝতে ছবি 3.3.2 লক্ষ করুন। ছবিটি একটি 2D গ্রাফ। আমরা আমাদের পিংজার উদাহরণে দুটি প্যারামিটার বা ফিচার ভ্যালু নিয়ে কাজ করেছিলাম, m ও c । এই দুটির মান পরিবর্তন করে করে আমরা অবশ্যে একটি বেস্ট ফিট লাইন পেয়েছিলাম, যেটি আমাদেরকে ভুলের মান সর্বনিম্ন দিত।

অধ্যায় ৩ : লিনিয়ার রিজেশন (Linear Regression)

এখন প্রেডিয়েন্ট ডিসেন্ট বোঝার সুবিধার্থে, আপাতত আমরা θ রে নিই আমাদের $c = 1$ । এটি পরিবর্তিত হবে না, একই থাকবে। পরিবর্তন হবে শুধু m -এর মান। এখন আমরা যদি এরকম একটি ফাংশন চিন্তা করি, যা ইনপুট হিসেবে নেবে m -এর মান এবং আউটপুট হিসেবে দেবে এর কিংবা কস্ট-এর মান; এবং আমরা এই মানগুলো দিয়ে যদি একটি 2D গ্রাফ আঁকি, যেখানে X অক্ষ বরাবর নেব m এবং Y অক্ষ বরাবর নেব Cost, তাহলে আমরা নিচের 2D গ্রাফগুলোর মতো একটি গ্রাফ পাব (গ্রাফ 3.3.2)। আমরা এখানে 4টি সেরকম 2D গ্রাফ নিয়েছি।

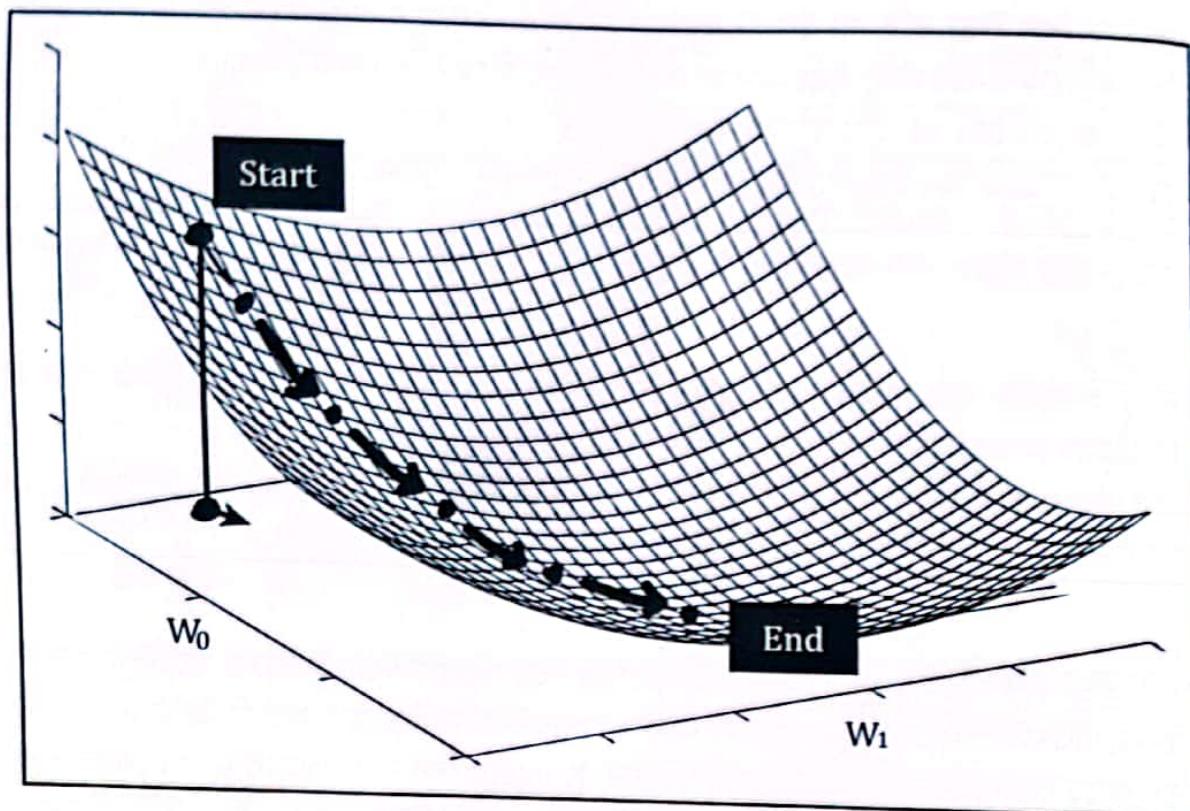


গ্রাফ 3.3.2

1, 2 ও 3 নম্বর গ্রাফে দেখুন, আমরা তিনটি ভিন্ন ভিন্ন পয়েন্ট (গোল কালো পয়েন্ট) স্পর্শক এঁকেছি। তিনটি স্পর্শকের ক্ষেত্রেই যদি আমরা স্পর্শকটির সঙ্গে X -অক্ষ কত ডিগ্রি কোণ উৎপন্ন করে আছে সেটি হিসাব করি এবং সেই কোণের Tangent বের করি, তাহলে দেখব কোনোটির মানই শূন্য হয় না। তাঁর মানে m -এর এই মানগুলোর জন্য আসলে ভুলের মান সর্বনিম্ন আসবে না। কিন্তু 4 নম্বর গ্রাফে দেখুন, আমরা m -এর যে মান নিয়েছি, সেই ডেটাপয়েন্টে যদি আমরা স্পর্শক আঁকি, তাহলে সেটি X -অক্ষের সমান্তরাল হয়। তাঁর মানে X -অক্ষের সঙ্গে স্পর্শকের কোণের মান 0 ডিগ্রি। অর্থাৎ, ঢাল হবে $\tan(0) = 0$ । অর্থাৎ, আমরা ঢালের মান সর্বনিম্নে নামিয়ে আনতে পেরেছি, তাঁর মানে এই পয়েন্টে কস্ট-এর মানও সর্বনিম্ন। কস্ট-এর মান 0 হবে – এরকম কোনো কথা নেই। কস্ট-এর মান সর্বনিম্ন হলেই হলো। আমরা তাই এভাবে, শুরুতে আন্দাজে (Randomly) m -এর যে-কোনো একটি মান ধরে নেব, এর পর ধাপে ধাপে m -এর মান আপডেট করতে করতে 4 নম্বর গ্রাফের মতো অবস্থায় নিয়ে আসব, যাতে আমরা সর্বনিম্ন কস্ট পাই।

আশা করি সবাই কিছুটা হলেও প্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদমের ধারণা বুঝতে পেরেছেন। এখানে আরেকটি গ্রাফ দেওয়া হলো (গ্রাফ 3.3.3)। এটি একটি 3D গ্রাফ। আগের বার আমরা c -এর মান ঠিক রেখে শুধু m -এর মান পরিবর্তন করেছিলাম। এবার আর তা নয়। m ও c দুটিই পরিবর্তিত হবে। তার মানে আমাদের ফিচার দাঁড়াল দুটি। আর সেই সঙ্গে আছে কস্ট। তাই আমাদের গ্রাফটি হচ্ছে 3D গ্রাফ। এই গ্রাফের X -অক্ষ ও Y -অক্ষ বরাবর আমরা নিয়েছি m ও c যাদেরকে এখানে W_0 ও W_1 হিসেবে চিহ্নিত করা হয়েছে। এর বিপরীতে, কস্ট বা এরর (error) প্লট করা হয়েছে Z -অক্ষ বরাবর, যাকে $E[W]$ দ্বারা চিহ্নিত করা হয়েছে। আমাদের লক্ষ্য হচ্ছে

আবারও আগের মতোই গ্রাফের সর্বনিম্ন বিন্দুতে পৌছানো, যেইখানে স্পর্শক অংকলে তাঁর ঢাকা এর মান শূন্য হবে।



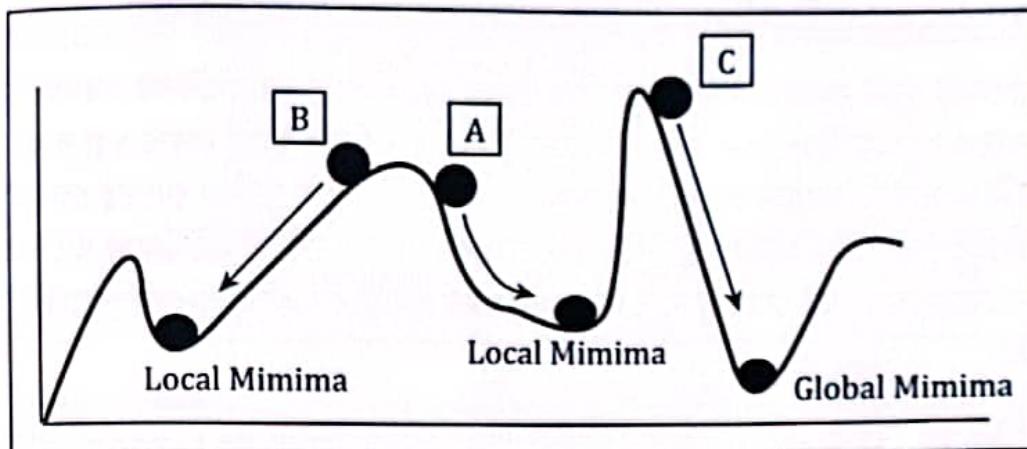
গ্রাফ 3.3.3

আমরা আবারও m ও c -এর ভ্যালু আগের মতোই আন্দাজে কোনো একটি মান দিয়ে শুরু করব, তারপর দুটির মানই আপডেট করতে থাকব, যতক্ষণ পর্যন্ত না গ্রাফের সর্বনিম্ন অবস্থানে পৌছাচ্ছি। এটিই গ্রেডিয়েন্ট ডিসেন্ট। গ্রাফ 3.3.3-তে দেখুন, আমরা Start লেখা পয়েন্ট থেকে শুরু করেছিলাম এবং দেখুন পয়েন্টটি কিন্তু আমাদের গ্রাউন্ড (এখানে গ্রাউন্ড বলতে আসলে গ্রাফ 3.3.3-এ দেখানো $W_0 - W_1$ তল বোঝানো হয়েছে) থেকে অনেক উচুতে, অর্থাৎ error অনেক বেশি। এরপর আমরা আন্তে আন্তে মান আপডেট করতে করতে End পয়েন্টে এসে পৌছেছি, যেটি একেবারে গ্রাউন্ডে, অর্থাৎ error-এর মান শূন্য। একটু ভালো করে লক্ষ করবেন, Start থেকে End-এ যাওয়ার ব্যাপারটি এখানে ছবি দেখে অনেকটা চাদরের গা বেয়ে স্লাইড করে নামার মতো মনে হচ্ছে এবং ব্যাপারটি খুবই মজার। ব্যাপারটি ঘটেও অনেকটা এরকমভাবেই। এই চাদরের মতো তলটিকে বলা হয় এরর প্লেন (Error Plane)।

এখানে আরেকটি বিষয় উল্লেখ্য। একটি গ্রাফে অনেকগুলো সর্বনিম্ন বিন্দু (Minimum Point) থাকতে পারে এবং সবগুলো বিন্দুতেই স্পর্শকের ঢালের মান শূন্য আসতে পারে। সেসব বিন্দুকে আমরা বলব লোকাল মিনিমা (Local Minima)। আর লোকাল মিনিমার মধ্যে যে লোকাল

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

মিনিমাম বিন্দু সর্বনিম্ন, তাকে বলব গ্লোবাল মিনিমাম (Global Minimum), যদি একাধিক গ্লোবাল মিনিমাম থাকে, তাহলে তাদের বলব গ্লোবাল মিনিমা (Global Minima)। নিচের ছবিটি দেখলে ধারণাটি আরেকটু পরিষ্কার হবে।



গ্রাফ 3.3.4

গ্রাফ 3.3.4-এ দেখুন, মোট তিনটি মিনিমাম পয়েন্ট আছে, এর মধ্যে একেবারে ডানে যেটি, সেটিই হচ্ছে আমাদের গ্লোবাল মিনিমাম, বাকি দুটি লোকাল মিনিমা। আমাদের সব সময় লক্ষ্য থাকবে যে, আমরা গ্লোবাল মিনিমাম পয়েন্টে পৌছাব।

একটি জিনিস লক্ষ্য করবেন, আমি যখন গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ব্যবহার করছি, তখন আমি শুধু এর প্লেন বেয়ে শুধু নিচের দিকে যেতে পারব, ওপরের দিকে কখনোই উঠতে পারব না। এর কারণ হচ্ছে, আমরা চাইছি কস্ট/এর মিনিমাইজ করতে, তাই না? আমরা যদি এর প্লেন বেয়ে ওপরের দিকে উঠি, তাহলে কিন্তু আমাদের এর বাড়তে থাকবে, যা আমাদের জন্য Allowed নয়। তাই, আমরা শুধু এর প্লেন বেয়ে নিচের দিকেই যেতে পারব, ওপরের দিকে উঠতে পারব না।

এখন ধরুন, আমরা এমনভাবে আমাদের m, c -এর মান ধরেছি, যে আমরা এর প্লেনে A অবস্থানে আছি। যদিও আমাদের এর প্লেনের গ্রাফটি 3D গ্রাফ ছিল, এখানে বোঝানোর সুবিধার জন্য 2D আকারে দেখাচ্ছি। আমাদের লক্ষ্য হচ্ছে A পয়েন্ট থেকে একেবারে ডান দিকে থাকা গ্লোবাল মিনিমাম পয়েন্টে আমাদের পৌছুতে হবে। এজন্য আমরা যেহেতু শুধু নিচের দিকে যেতে পারব, তাই ডান দিকের ঢাল বেয়ে নিচের দিকে নামতে থাকি (অনেকটা পাহাড় থেকে নামার মতো)। নামতে নামতে একসময় দেখুন আমরা একটি লোকাল মিনিমাম পয়েন্টে এসে আটকা পড়ে যাব!

কেননা, আমাদের যদি গ্লোবাল মিনিমামে পৌছুতে হয়, তাহলে আমাদের লোকাল মিনিমান পার হয়ে পাহাড় বেয়ে ওপরে উঠে আবার নিচে নামতে হবে, তবেই আমরা সেখানে পৌছুতে পারব। কিন্তু, পাহাড় বেয়ে ওপরে ওঠা আমাদের জন্য Allowed নয়, সেটি আগেই বলা হয়েছে। তাহলে

ব্যাপারটি কী দাঁড়াল? আমরা লোকাল মিনিমান পয়েন্টে আটকা পড়ে গেলাম এবং এখান থেকে আমরা আর বের হতে পারব না, আমাদের অ্যালগরিদম এটিকেই আমাদের সমাধান হিসেবে রিটার্ন করবে যদিও এটি মোটেই গ্লোবাল মিনিমাম নয়। এই ধরনের পরিস্থিতিকে বলা হয় ‘Stuck in Local Minimum’, যেটি মেশিন লার্নিংয়ের ক্ষেত্রে খুবই সাধারণ একটি সমস্যা।

এর সমাধানটি খুবই সহজ। আমাদের আবার গোড়া থেকে পুরো অ্যালগরিদম শুরু করতে হবে এবং এবারও র্যান্ডমলি m, c -এর অন্য আরেকটি মান ধরে নিতে হবে। এবার ধরা যাক, আমরা র্যান্ডমলি আবার B অবস্থান থেকে শুরু করলাম। এবারও দেখুন, আমরা যেহেতু কেবল নিচের দিকে যেতে পারি, তাই আমরা বাঁ দিকে যেতে থাকব, যার ফলে আমরা ধাপে ধাপে গ্লোবাল মিনিমাম থেকে দূরে সরে যেতে থাকব এবং একসময় একটি লোকাল মিনিমান পয়েন্টে আটকা পড়ে যাব।

তখন আমাদের আবার শুরু থেকে শুরু করতে হবে। এবার আমরা m, c -এর মান র্যান্ডমলি নিয়ে, ধরা যাক, C অবস্থান থেকে শুরু করি, তাহলে দেখুন এবার কিন্তু আমরা ডান দিক দিয়ে নিচে নামতে নামতে গ্লোবাল মিনিমাম পয়েন্টে পৌঁছে যাব। এই পদ্ধতিকে বলা হয় Random Restart এবং এটি ততক্ষণ পর্যন্ত আমরা করতে থাকব যতক্ষণ পর্যন্ত না আমরা গ্লোবাল মিনিমান পয়েন্টে পৌঁছাচ্ছি কিংবা আমাদের ইটারেশন থ্রেশহোল্ড (Iteration Threshold – সর্বোচ্চ কর্তব্য Global Minimum খুঁজে না পাওয়া পর্যন্ত Random Restart করতে পারব তার একটি মান নির্ধারিত করে দেওয়া থাকে) অতিক্রম করে যাচ্ছি।

এখন কথা হচ্ছে, অ্যালগরিদমের তো জানার কথা নয় যে Global বা Local Minimum-এর মান কত, তাহলে সে কেমন করে বুঝবে যে সে মিনিমামে পৌঁছেছে কি না। এর উত্তর হচ্ছে, আমরা প্রতিবার লুপ ঘুরে আসার পরে যাচাই করে দেখব যে, আমরা বর্তমান m, c -এর জন্য Error-এর যে মান পেলাম, সেটি আগে যতবার লুপ ঘুরেছে প্রতিবারের পাওয়া সব Error-এর মানের মধ্যে সর্বনিম্ন কি না, যদি হয় তাহলে এটি মিনিমাম মান হিসেবে সংরক্ষণ করব, আর যদি না হয়, তাহলে এবারের আগ পর্যন্ত মিনিমাম মান যা ছিল তা-ই থাকবে এবং পরের বার লুপ চালাব। এভাবে একসময় লুপ চলতে চলতে যদি আমরা দেখি যে বেশ অনেকবার লুপ চলেছে আর আমাদের থ্রেশহোল্ডও পূর্ণ হয়ে গেছে কিন্তু Error-এর মান আর কমছে না, তখন ধরে নেব Error-এর মান এর চেয়ে আর কমা সন্তুষ্ট নয়। তখন তাকেই আমরা সর্বনিম্ন মান হিসেবে ধরে নেব।

এখন, এই সর্বনিম্ন মান Global/Local যে-কোনোটিই হতে পারে। তাই, তাকে আবার যাচাই করে নেওয়ার জন্য Randomly আরো বেশ কয়েকবার Restart করে পুরো লুপ চালাব। ধরলৈ আমি 3 বার Random Restart করলাম, প্রতিবার আমি 10 বার করে লুপ চালালাম। প্রথম Random Restart-এ আমি এররের মান পেলাম ধরা যাক 10, দ্বিতীয়বার পেলাম 8 এবং তৃতীয়বার পেলাম 4। সুতরাং, এই তিনটির মধ্যে সর্বনিম্ন হচ্ছে 4, আর তাই তৃতীয় Random

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

Restart-এ, যে m, c -এর জন্য আমরা error হিসেবে ৪ পেয়েছি, সেই m, c -এর মানই হবে আমার বেস্ট ফিট লাইনের m, c -এর মান।

এই হচ্ছে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম সম্পর্কে যতটুকু জানার দরকার তার বিস্তারিত। পরবর্তী সময়ে আমরা দেখব কীভাবে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম বিভিন্ন মেশিন লার্নিং পদ্ধতিতে ব্যবহার করতে হয়।

পরিচেদ ৩.৪ : গ্রেডিয়েন্ট ডিসেন্ট (Gradient Descent) অ্যালগরিদম ব্যবহার করে লিনিয়ার রিগ্রেশন

আমরা প্রথমে দুটি ফাংশন নির্ধারণ করব, একটি হচ্ছে কস্ট ফাংশন (Cost Function), আরেকটি অবজেক্টিভ ফাংশন (Objective Function)। আমাদের প্রথমে বুঝতে হবে এই দুটি ফাংশন আসলে কী।

ধোঁযাক, আমার বয়স 20 বছর। আমি একটি প্রোগ্রাম লিখলাম, যার কাজ হচ্ছে আমার বয়স আন্দাজ করা এবং যদি সে আমার ঠিক বয়স বলতে পারে, তাহলে প্রোগ্রাম বন্ধ হয়ে যাবে। আমার আসল বয়সকে যদি আমি A দিয়ে লিখি এবং আমার আন্দাজ করা বয়সকে যদি আমি X দিয়ে লিখি, তাহলে একটি কস্ট ফাংশন এরকম হতে পারে $F(X) = X - A$ । এটি সরাসরি আমার আসল বয়স এবং প্রোগ্রামের দ্বারা আন্দাজ করা বয়সের পার্থক্য রিটার্ন করবে।

আর অবজেক্টিভ ফাংশন হচ্ছে, যে ফাংশনটিকে আমরা অপটিমাইজ করতে চাইছি, সেটিই। অর্থাৎ, আমাদের এই ক্ষেত্রে, আমরা চাইছি যে আমাদের কস্ট কমিয়ে আনতে, অর্থাৎ X -এর এমন মান আন্দাজ করতে চাইছি, যাতে সেটি আমার আসল বয়সের সমান হয়, ফলে কস্ট-এর মান শূন্য হয়। সাধারণত মেশিন লার্নিংয়ে এই কস্ট ও অবজেক্টিভ ফাংশন একই অর্থে ঘুরিয়ে-ফিরিয়ে ব্যবহার করা হয়।

কস্ট ফাংশন অনেক ধরনের হতে পারে। একেকটি ব্যবহারের একেক রকম সুবিধা। আমরা লিনিয়ার রিগ্রেশনের ক্ষেত্রে সাধারণত যে ফাংশনটি কস্ট ফাংশন হিসেবে ব্যবহার করব, তার নাম Squared Error Function। এটি খুবই জনপ্রিয় এবং খুব সন্তুষ্ট সবচেয়ে বেশি ব্যবহৃত একটি কস্ট ফাংশন।

কস্ট ফাংশনটি লেখার আগে আমরা একটি কাজ করব, তা হলো, c ও m -কে এখন থেকে আমরা w_0 ও w_1 দিয়ে প্রকাশ করব। এই w_0, w_1 ইত্যাদিকে ওয়েইট (Weight) বলা হয়। এর কারণ হচ্ছে, এদেরকে যে রাশির সঙ্গে সহগ বা কো-এফিসিয়েন্ট (Co-efficient) হিসেবে ব্যবহার করা হয়, এরা সেই সংখ্যার গুরুত্ব প্রকাশ করে।

আর আমাদের পরিচ্ছেদ ৩.২-এ ব্যবহৃত হাইপোথিসিসের সমীকরণে আমরা ফাইনাল আউটপুট বোঝানোর জন্য Y ব্যবহার করেছিলাম, এখন Y -এর পরিবর্তে $h_w(x)$ ব্যবহার করব। $h_w(x)$ দিয়ে বোঝাবে আমরা x ডেটা পয়েন্টের জন্য কাজ করছি এবং আমাদের প্যারামিটার (চাল w ও b , intercept) হচ্ছে w সেটের অন্তর্ভুক্ত (w_0 ও w_1)। আমরা এই কাজটি করব আমাদের কিছু গাণিতিক সমীকরণ লেখার সুবিধার্থে। অর্থ একই থাকবে, সমীকরণ একই থাকবে, শুধু প্রকাশকালীন চিহ্নগুলো পরিবর্তন করলাম মাত্র।

তাহলে আমাদের হাইপোথিসিসের সমীকরণের চেহারা দাঁড়ায়,

$$h_w(x) = w_0 + w_1 x$$

তার মানে $h_w(x)$ -এর মান হচ্ছে আমাদের প্রেডিকটেড (Predicted) বা অনুমিত মান, যা কিনা আমরা w_0 ও w_1 -এর মান বের করার মাধ্যমে হিসাব করে বের করছি। আর Y -কে যদি আমরা আমাদের আসল আউটপুটের মান ধরে নিই, যেটি আমাদের লেবেলড ট্রেনিং ডেটা হিসেবে দেওয়া আছে, তাহলে আমাদের এরর কিংবা কস্ট ফাংশন-এর মান প্রাথমিক রূপ দাঁড়ায়,

$$h_w(x) - Y$$

এখন দেখুন, আমরা যদি আমাদের কস্ট ফাংশনকে এমন একটি অবজেকটিভ ফাংশন হিসেবে লেখি, যাতে আমাদের কস্ট ফাংশনকে মিনিমাইন করতে হয়, তাহলে,

$$\text{Cost} = h_w(x) - Y$$

এ ক্ষেত্রে, $h_w(x)$ -এর মান যত বেশি $-\infty$ (negative infinity)-এর দিকে যাবে, অর্থাৎ যত বেশি ঋণাত্মক হবে, তত কস্ট ফাংশনের মান মিনিমাইজড হবে। কিন্তু দেখুন, শুধু কস্ট ফাংশন মিনিমাইজ করলেই কি আমাদের কাজ হাসিল হচ্ছে? আমাদের লক্ষ্য হচ্ছে আমাদের আন্দাজ করা মান $h_w(x)$ ও আসল মান Y -এর সমান করা, কিংবা যতটা পারা যায় কাছাকাছি নিয়ে আসা এবং তার মাধ্যমে কস্ট ফাংশনকে মিনিমাইজ করা। কিন্তু যদি, আমরা শুধু $h_w(x)$ -এর মান $-\infty$ -এর দিকে নিতে থাকি, তাহলে কস্ট ফাংশনের মান মিনিমাইজ হচ্ছে ঠিকই, কিন্তু দুটি মান কাছাকাছি আসবে বা সমান হবে, এরকম কোনো বাধ্যবাধকতা (Constraint) তৈরি হচ্ছে না।

তাই, আমরা কস্ট ফাংশন হিসেবে শুধু এরর ফাংশন না নিয়ে স্ক্যারড এরর ফাংশন (Squared Error Function) নেব,

$$\{h_w(x) - y\}^2$$

এর ফলে একটি সুবিধা হবে যে, এই ফাংশনের মান তখনই কেবল মিনিমাইজ হবে যখন $h_w(x)$ ও Y সমান বা খুব কাছাকাছি হবে। যদি $h_w(x)$ ও Y -এর মান সমান হয়, তাহলে, কস্ট ফাংশনের মান হবে শূন্য, যেটি এর সর্বনিম্ন মান। $h_w(x)$ -এর মান আমরা এখন যতই ঋণাত্মক বানাই না

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

কেন, শেষমেশ সেটি বর্গ হয়ে একটি বড়ো ধনাত্মক মানে পরিণত হবে, যেটি আমাদের কস্ট ফাংশন মিনিমাইজ করার লক্ষ্য পরিপূর্ণ করবে না।

এভাবে Squared Error Function ব্যবহার করলে যেহেতু আমরা দুটো মান সমান হওয়ার বাধ্যবাধকতাও দিতে পারছি এবং সেই সঙ্গে কস্ট ফাংশনও মিনিমাইজ করতে পারছি, এ কারণে Squared Error ফাংশনকেই আমরা কস্ট ফাংশন হিসেবে ব্যবহার করব।

এখন, কস্ট ফাংশন হিসেবে যদি শুধু $\{h_w(x) - y\}^2$ রাশিটি নিই, তাহলে আমরা যে কস্ট পাব, সেটি হচ্ছে আমাদের একটিমাত্র সিংগেল ডেটা পয়েন্ট কিংবা ট্রেনিং ডেটা x (একটিমাত্র পিংজার সাইজ)-এর জন্য। কিন্তু আমাদের লক্ষ্য হচ্ছে সবগুলো ট্রেনিং ডেটার জন্য আমাদের একটি সামগ্রিক কস্ট হিসাব করা। সেটি যদি করতে হয়, তাহলে আমাদের সব ডেটা পয়েন্টের জন্য এরের হিসাব করে তাকে মোট ট্রেনিং ডেটার সংখ্যা দিয়ে ভাগ করে গড় মানটি নিতে হবে। যদি তা-ই করি, তাহলে আমাদের কস্ট ফাংশনের সর্বশেষ চেহারা দাঁড়াবে :

$$= \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

এখানে $x^{(i)}$, $y^{(i)}$ মানে হচ্ছে i -তম ট্রেনিং ডেটা এবং m হচ্ছে মোট ট্রেনিং ডেটার সংখ্যা। এ ছাড়া এখানে একটি অতিরিক্ত 2 দিয়ে ভাগ করা হয়েছে কিছু গাণিতিক হিসাবের সুবিধার্থে।

কস্ট ফাংশনটিকে আমরা যদি $J(w_0, w_1)$ দিয়ে প্রকাশ করি (শুধুই একটি গাণিতিক প্রকাশ), তাহলে আমাদের ফাংশনের সর্বশেষ রূপ দাঁড়ায় এরকম –

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

এটিই আমাদের কাঞ্জিক্ত কস্ট ফাংশন। আমাদের লক্ষ্য হচ্ছে এই ফাংশনটির প্যারামিটার w_0 ও w_1 -এর এমন কোনো একটি মান বের করা, যার জন্য $J(w_0, w_1)$ -এর মান সর্বনিম্নে নামিয়ে নিয়ে আসা (মিনিমাইজ করা) যায়। এটিই আমাদের অবজেক্টিভ ফাংশন এবং এটিকে লেখা যায় এভাবে :

$$\begin{aligned} \text{Objective Function} &= \underbrace{\min_{w_0, w_1}}_{w_0, w_1} J(w_0, w_1) \\ &= \underbrace{\min_{w_0, w_1}}_{w_0, w_1} \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 \end{aligned}$$

এখন আমাদের এই লিনিয়ার রিগ্রেশন চালানোর জন্য প্রেডিয়েন্ট ডিসেন্টের ধাপগুলো দাঁড়াচ্ছে :

- প্রথমে w_0 ও w_1 -কে দুটি র্যান্ডম মান অ্যাসাইন করি (আপনারা দুটির মানই 1 দিয়ে শুরু করতে পারেন)।

মেশিন লার্নিং অ্যালগরিদম

- এরপর w_0 ও w_1 -এর মান প্রতি ইটারেশনে একবার করে আপডেট করতে হবে এমনভাবে যাতে $J(w_0, w_1)$ -এর মান ক্রমশ কমে আসতে থাকে।
- একসময় দেখা যাবে 5-10 ইটারেশনের পরেও $J(w_0, w_1)$ -এর মান আর উল্লেখযোগ্য হারে কমছে না। অর্থাৎ লুপের ইটারেশন সংখ্যার বিপরীতে $J(w_0, w_1)$ প্লট করলে, দেখা যাবে ইটারেশনের সংখ্যা একটি মান অভিক্রম করার পর গ্রাফ এক জায়গায় প্রায় সরলরৈখিক হয়ে গেছে। হয়তো 4-5 বার লুপ ঘোরালে 0.001 কমছে কিংবা এরকম খুব অল্প পরিমাণে কমছে। তখন আপনি আপনার লুপ চালানো শেষ করে দিতে পারেন।
- লুপ শেষ হয়ে গেলে আপনি যে w_0 ও w_1 -এর মানগুলো পেলেন, এটিই এদের অপটিমাল ভ্যালু, যার জন্য আপনি $J(w_0, w_1)$ -এর মান সর্বনিম্ন পাচ্ছেন।
- এখন w_0 ও w_1 মান আপনি আপনার হাইপোথিসিস $h_w(x) = w_0 + w_1x$, সমীকরণে বসালেই আপনার কাঙ্ক্ষিত হাইপোথিসিস পেয়ে যাবেন। এখন এতে কোনো অজানা x -এর মান বসালে আমরা তার জন্য কাঙ্ক্ষিত আউটপুট পেয়ে যাব।

সবশেষে যেটি বাকি থাকে, সেটি হচ্ছে আমরা w_0 ও w_1 -এই প্যারামিটার দুটির মান আপডেট কীভাবে করব? এই আপডেটের অংশটুকুই হচ্ছে আমাদের কাজের মূল অংশ।

এর জন্য সোজাসাপটা সূত্র হচ্ছে –

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1)$$

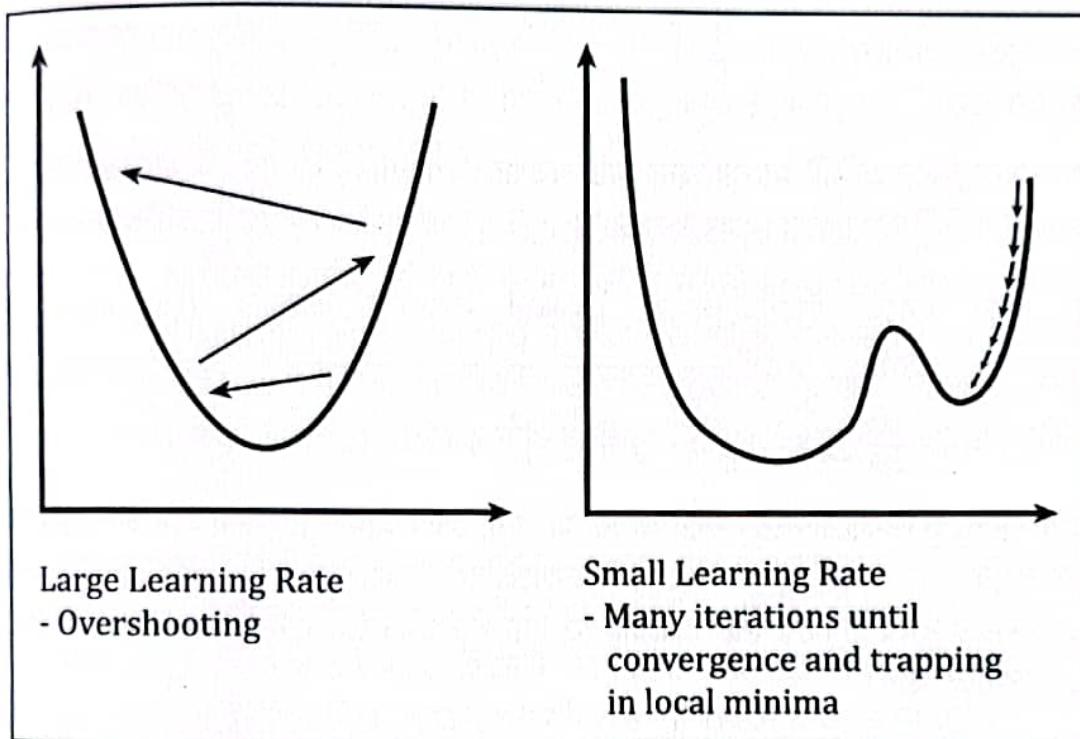
দেখতে অনেক নিরীহ মনে হলেও, আসলে অতটা সরল নয় এটি। কিছু জটিলতা আছে এর মধ্যে। সমীকরণটির প্রতিটি অংশের কোনটির কী অর্থ, তাহলে সেটি একটু জেনে নেওয়া যাক।

প্রথমেই জেনে নিই, সমীকরণটি ব্যবহৃত হচ্ছে j -এর সব মানের জন্য। এখানে j -এর মান হচ্ছে 0, 1 ইত্যাদি যা আমরা w_0, w_1 ইত্যাদি লেখার সময় সাবসক্রিপ্ট হিসেবে ব্যবহার করেছি। আমরা যদি আরো একটি ভ্যারিয়েবল নিতাম (হয়তো পিংজার সাইজের পাশাপাশি অন্য কোনো একটি ফিচার) এবং তাকে হয়তো x_2 দিয়ে প্রকাশ করে তাঁর জন্য ওয়েইট w_2 ব্যবহার করতাম, তাহলে j -এর মান হতো 0, 1, 2 ইত্যাদি পর্যন্ত।

এর পরে, α বলতে এখানে বোঝায় লার্নিং রেট কিংবা স্টেপ সাইজ। α -এর মান যত বড়ো হবে, আমাদের লিনিয়ার রিঞ্চেশন অ্যালগরিদম তত বড়ো বড়ো মান যোগ/বিয়োগ করে w_0, w_1 -এর মান আপডেট করবে। এর মান ঠিকমতো নেওয়ার বিশাল গুরুত্ব আছে। যদি α -এর মান অনেক বেশি বড়ো নিই, তাহলে অ্যালগরিদম আমাদের $J(w_0, w_1)$ -এর সর্বনিম্ন মানের দিকে আগাবে অনেক দ্রুত, কিন্তু বড়ো বড়ো মান যোগ/বিয়োগ করে আপডেট করার কারণে (বড়ো বড়ো স্টেপ নেওয়ার কারণে) অনেক সময় আমরা সর্বনিম্ন $J(w_0, w_1)$ পয়েন্টটি পার হয়ে যেতে পারি (গ্রাফ

অধ্যায় ৩ : লিনিয়ার রিট্রেশন (Linear Regression)

৩.৪.১-এর বাঁয়ের ছবি। আবার যদি α -এর মান অনেক বেশি ছোটো নিই, তাহলে অ্যালগরিদম কনভার্জ (Converge) করতে অনেক বেশি সময় লেবে। অর্থাৎ সর্বনিম্ন $J(w_0, w_1)$ -তে পৌছাতে অনেক বেশি ইটারেশন লাগবে (গ্রাফ ৩.৪.১-এর ডানের ছবি)। তাই α -এর মান খুব সাবধানে পছন্দ করতে হবে, যেন বেশি বড়ো না হয়, ছোটোও না হয়। এটি সাধারণত 0.01 রাখা হয়ে থাকে।



ছবি ৩.৪.১

গ্রেডিয়েন্ট ডিসেন্টকে আরেকটু দ্রুততর করার জন্য এক ধরনের অপটিমাইজেশন চালানো হয়। এর পেছনে মূল ধারণাটি হচ্ছে, এটি গ্লোবাল মিনিমান থেকে যত দূরে থাকবে, লার্নিং রেট α -এর মান তত বড়ো থাকবে, সে যত গ্লোবাল মিনিমামের কাছাকাছি আসতে থাকবে, α -এর মান তত কমবে।

এখন, যেহেতু আপনি জানেন না যে গ্লোবাল মিনিমাম মানে আপনি আদৌ পৌছেছেন কি না, তাই এ ক্ষেত্রে একটি পদ্ধতি প্রয়োগ করা হয়। প্রতিবার লুপ চলার পরে, একবার করে error-এর মান পরীক্ষা করে দেখবেন। যদি দেখেন এরর আগের বারের চেয়ে কিছুটা কমেছে (10^{-10} -এর চেয়ে বেশি কমেছে), তাহলে α -এর মান 5% বাঢ়াবেন। অর্থাৎ α হয়ে যাবে 1.05α ।

আবার যদি দেখেন, এরর আগের চেয়ে তো কমেইনি বরং বেড়েছে, সে ক্ষেত্রে α -এর মান হিসেবে শেষবার আপডেট করার আগে α -এর মান যা ছিল তার 50% নেবেন, অর্থাৎ α হয়ে যাবে 0.5α । এই অপটিমাইজেশন পদ্ধতির নাম হচ্ছে Bold Driver। এরকম আরো কিছু অপটিমাইজেশন

আছে, যেমন – Adagrad, Adam, Adamax, RMSProp ইত্যাদি, যেগুলো আপনারা চাইলে
এই লিংক থেকে দেখে নিতে পারেন – <http://ruder.io/optimizing-gradient-descent/>

সবশেষে $\frac{\partial y}{\partial x}$ দিয়ে বোঝায় x -এর সাপেক্ষে y -এর পার্শ্বিয়াল ডেরিভেটিভ। আমরা আমাদের
সমীকরণে ব্যবহার করেছি $\frac{\partial}{\partial w_j} J(w_0, w_1)$ ।

$$\text{সুতরাং এখানে } x = w_j \text{ ও} \\ y = J(w_0, w_1)$$

অর্থাৎ আমরা j -এর প্রতিটি মানের জন্য মোট যতগুলো প্যারামিটার পাব (আমাদের ক্ষেত্রে 2টি,
 w_0 ও w_1) প্রতিটির সাপেক্ষে একবার করে $J(w_0, w_1)$ -এর পার্শ্বিয়াল ডেরিভেটিভ নেব।

সোজা করে বলতে গেলে, আমরা w_0 -এর সাপেক্ষে একবার $J(w_0, w_1)$ অর্থাৎ
 $\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$ -কে ডিফারেন্সিয়েট করব, তারপরে w_1 -এর সাপেক্ষে আবার
 $J(w_0, w_1)$ অর্থাৎ $\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$ -কে ডিফারেন্সিয়েট করব।

w_0 -এর সাপেক্ষে ডিফারেন্সিয়েট করে আমরা যা পাব, সেটি আমরা w_0 -এর মান আপডেট করার
সময় ব্যবহার করব। আর w_1 -এর সাপেক্ষে ডিফারেন্সিয়েট করে আমরা যা পাব তা w_1 -এর মান
আপডেট করার সময় ব্যবহার করব। প্রতিটি আলাদা প্যারামিটারের জন্য এভাবে আলাদা আলাদা
আপডেট করতে হবে।

এখন, আমরা যদি, w_0 -এর সাপেক্ষে একবার $\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$ -কে
ডিফারেন্সিয়েট করি, তাহলে আমরা পাই –

$$\begin{aligned} \frac{\partial}{\partial w_0} J(w_0, w_1) &= \frac{\partial}{\partial w_0} \left[\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 \right] \\ &= \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \end{aligned}$$

একইভাবে, আমরা যদি, w_1 -এর সাপেক্ষে একবার $\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$ -কে
ডিফারেন্সিয়েট করি, তাহলে আমরা পাই,

$$\begin{aligned} \frac{\partial}{\partial w_1} J(w_0, w_1) &= \frac{\partial}{\partial w_1} \left[\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 \right] \\ &= \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x^{(i)} \end{aligned}$$

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

যাঁদের মোটামুটি ক্যালকুলাসে হাত ঢালু আছে, তাঁরা ঢাইলে হিসাব কষে মিলিয়ে দেখতে পারেন হিসাবটি এরকমই কি না। আমি সরাসরি উভয় দিয়ে দিলাম, যেহেতু আগেই বলেছি আমরা থিওরি আর কঠিন কঠিন অঙ্ক যতটুকু লাগে ঠিক ততটুকুই শিখব। যা-ই হোক, তাহলে আমরা w_0 ও w_1 এই দুটি প্যারামিটারের জন্য এই দুটি সমীকরণ ব্যবহার করব।

এখানে একটি বিষয় খেয়াল করে দেখবেন, প্রতিবার আমরা w_0 ও w_1 আপডেট করার সময় সমস্ত ট্রেনিং ডেটা ব্যবহার করছি। $\frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}$ থেকে দেখতে পাচ্ছি আমরা এখানে সমস্ত ট্রেনিং ডেটার ওপরে Error বের করে নিচ্ছি। এই ধরনের গ্রেডিয়েন্ট ডিসেন্টকে অনেক সময় Batch Gradient Descent বলে।

এর আরেকটি ভার্শন আছে, যেখানে সমস্ত ট্রেনিং ডেটা ব্যবহার করার বদলে শুধু যে ট্রেনিং ডেটার জন্য w_0 ও w_1 আপডেট করছি, শুধু সেই ট্রেনিং ডেটার জন্যই error বের করে ব্যবহার করে থাকি, অর্থাৎ শুধু $\{h_w(x^{(i)}) - y^{(i)}\}$ এতটুকু ব্যবহার করে থাকি, বাকি সব এক। সেটাকে বলা হয় Stochastic Gradient Descent। আপনারা যে-কোনোটিই ব্যবহার করতে পারেন, তবে আমি এই বইতে থিওরি বোঝানোর সময় ব্যাচ গ্রেডিয়েন্ট ডিসেন্টই ব্যবহার করব বেশিরভাগ সময়।

তাহলে এই হচ্ছে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ব্যবহার করে কীভাবে আমরা লিনিয়ার রিগ্রেশন করতে পারি, তার বিস্তারিত বর্ণনা। আশা করি সবাই বুঝতে পেরেছেন। যাঁদের এটি বুঝতে সমস্যা হচ্ছে এখনো, তাঁরা পরিচ্ছেদ ৩.৮-এ গ্রেডিয়েন্ট ডিসেন্ট ব্যবহার করে কীভাবে লিনিয়ার রিগ্রেশন করতে হয়, সে বর্ণনাটি দেখে নিতে পারেন পরবর্তী ৩.৫ পরিচ্ছেদে যাওয়ার আগেই।

পরিচ্ছেদ ৩.৫ : গ্রেডিয়েন্ট ডিসেন্ট (Gradient Descent) অ্যালগরিদম ব্যবহার করে মাল্টিভ্যারিয়েট (Multivariate) লিনিয়ার রিগ্রেশন

আমরা এতক্ষণ যে লিনিয়ার রিগ্রেশন করলাম, সেটি ছিল ইউনিভ্যারিয়েট (Univariate) লিনিয়ার রিগ্রেশন। এর অর্থ হচ্ছে, আমাদের ফিচার/ভ্যারিয়েবল ছিল মাত্র একটি (পিংজার সাইজ) যাকে আমরা শুধু x দিয়ে প্রকাশ করতাম এবং সেটি ব্যবহার করে আমরা পিংজার দাম আন্দাজ করতাম।

এখন আমি যদি আরেকটি উদাহরণ নিই, যেখানে আমাদের একাধিক ফিচার আছে। যেরকম, আমরা যদি টেবিল ৩.৫.১ দেখি যেখানে কতগুলো ফোনের দাম দেওয়া আছে এবং সঙ্গে তার কিছু ফিচারের মান লেখা আছে (রঞ্জ, রঞ্জ ইত্যাদি)।

মেশিন লার্নিং অ্যালগরিদম

র্যাম (GB)	রম (GB)	ক্যামেরা (MP)	ডিসপ্লে সাইজ (inch)	দাম (BDT)
2	16	8	5.0	10,000
4	32	8	5.2	12,500
6	64	12	5.5	15,000

টেবিল 3.5.1

এই টেবিল থেকে আমরা দেখতে পাচ্ছি, আমাদের ফিচারের সংখ্যা 4টি – র্যাম, রম, ক্যামেরা এবং ডিসপ্লে সাইজ। আর দাম হচ্ছে আমাদের আউটপুট। ফিচারগুলোকে আমরা এখন x_1, x_2, x_3 ও x_4 দিয়ে প্রকাশ করতে পারি। এদের জন্য Weight হবে তাহলে যথাক্রমে w_1, w_2, w_3 ও w_4 ।

এখন আমরা যদি এই একাধিক ফিচার ব্যবহার করে লিনিয়ার রিপ্রেশন করে দাম জানা নেই, এরকম কোনো মোবাইলের ফিচারের মানগুলো আগের মতোই হাইপোথিসিসে বসিয়ে সেই মোবাইলের দাম বের করি, তাহলে সেই লিনিয়ার রিপ্রেশনকে বলা হবে মাল্টিভ্যারিয়েট (Multivariate) লিনিয়ার রিপ্রেশন।

এখন তাহলে মাল্টিভ্যারিয়েট লিনিয়ার রিপ্রেশনের হাইপোথিসিস সমীকরণটি লিখে ফেলা যাক। এটি আমাদের আগের ইউনিভ্যারিয়েট লিনিয়ার রিপ্রেশনের সমীকরণের সঙ্গে অনেকখানিই সামঞ্জস্যপূর্ণ :

$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$$

এখন আমরা যদি $x_0 = 1$ ধরি, তাহলে আমরা আমাদের সমীকরণ দাঁড়ায়,

$$h_w(x) = w_0x_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$$

একে জেনারালাইজ করে n -সংখ্যক ফিচারের জন্য লিখলে পাই,

$$h_w(x) = w_0x_0 + w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n$$

আমরা এখন w নামে একটি ওয়েইট ভেক্টর (Weight Vector) চিন্তা করি, যেখানে $w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix}$

অধ্যায় ৩ : লিনিয়ার রিপ্রেশন (Linear Regression)

এবং x নামে আরেকটি ফিচার ভেস্টর (Feature Vector) চিন্তা করি, যেখানে $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$

এখন যদি আমরা $w^T x$ নিই, তাহলে আমরা পাই, $w^T x = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix}^T \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$

এখনে উল্লেখ্য, যে-কোনো ম্যাট্রিক্স A -এর ট্রান্সপোজ করলে ওই ম্যাট্রিক্স-এর রো (Row)-গুলো কলাম (Column) এবং কলামগুলো রো-তে রূপান্তরিত হয়ে যায় এবং একে A^T দিয়ে প্রকাশ করা হয়।

$$\text{সূত্রাঃ } w^T x = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix}^T \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

$$= [w_0 \ w_1 \ w_2 \ \dots \ w_n] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

$$= w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \dots + w_n x_n \quad (\text{ম্যাট্রিক্স গুণনের নিয়ম অনুসারে})$$

সূত্রাঃ আমরা এখন লিখতে পারি যে,

$$h_w(x) = w^T x$$

এটি আমাদের মাল্টিভ্যারিয়েট লিনিয়ার রিপ্রেশনের হাইপোথিসিস।

আমাদের কস্ট ফাংশন এবং অবজেকটিভ ফাংশন আগের মতোই থাকবে।

কস্ট ফাংশনে সামান্য পরিবর্তন যেটি আসবে যে আমরা এখন w_n পর্যন্ত লিখব

$$J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

$$\text{আর, } \text{Objective Function} = \underbrace{\min_{w_0, w_1, w_2, \dots, w_n}}_{w_0, w_1, w_2, \dots, w_n} J(w_0, w_1, w_2, \dots, w_n)$$

$$= \underbrace{\min_{w_0, w_1, w_2, \dots, w_n}}_{w_0, w_1, w_2, \dots, w_n} \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

তাহলে এখন আগের মতোই প্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম দিয়ে প্যারামিটার আপডেটের মাধ্যমে মাল্টিভ্যারিয়েট লিনিয়ার রিগ্রেশন চালাই, তাহলে আমাদের কাজের ধাপগুলো দাঁড়াচ্ছে -

- প্রথমে $w_0, w_1, w_2, \dots, w_n$ প্রত্যেককে র্যান্ডম মান অ্যাসাইন করি (আপনারা আগের মতোই সবগুলোর মান 1 দিয়ে শুরু করতে পারেন)
- এরপর $w_0, w_1, w_2, \dots, w_n$ -এর মান প্রতি ইটারেশনে একবার করে আপডেট করতে হবে এমনভাবে, যাতে $J(w_0, w_1, w_2, \dots, w_n)$ -এর মান কমে আসতে থাকে।
- আগের মতোই একটি সময় গিয়ে দেখা যাবে 5-10 ইটারেশনের পরেও $J(w_0, w_1, w_2, \dots, w_n)$ -এর মান আর উল্লেখযোগ্য হারে কমছে না। তখন আপনি আপনার লুপ চালানো শেষ করে দিতে পারেন।
- লুপ শেষ হয়ে গেলে আপনি $w_0, w_1, w_2, \dots, w_n$ -এর যে মানগুলো পেলেন, এটিই এদের অপটিমাল মান, যার জন্য আপনি $J(w_0, w_1, w_2, \dots, w_n)$ -এর মান সর্বনিম্ন পাচ্ছেন।
- এখন $w_0, w_1, w_2, \dots, w_n$ এদের মান হাইপোথিসিস $h_w(x) = w^T x$ -এ বসালেই আপনি আপনার কানিক্ষিত হাইপোথিসিস পেয়ে যাবেন। এখন এতে কোনো অজানা $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ -এর মান বসালেই আমরা তার জন্য কানিক্ষিত আউটপুট কী হবে তা পেয়ে যাব।

সবশেষে যেটি বাকি থাকে, সেটি হচ্ছে আমরা $w_0, w_1, w_2, \dots, w_n$ এই প্যারামিটারগুলোর মান আপডেট কীভাবে করব? আমরা সেই আগের অ্যালগরিদমই অনুসরণ করব -

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1, w_2, \dots, w_n)$$

আমাদেরকে $\frac{\partial}{\partial w_j} J(w_0, w_1, w_2, \dots, w_n)$ মান সেই আগের নিয়মেই বের করতে হবে। এর জন্য একটি সাধারণ সূত্র আছে -

$$\frac{\partial}{\partial w_j} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_j^{(i)}$$

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

এখানে x_j দ্বারা বোঝায় ট্রেনিং ডেটার j -তম ফিচার। সুতরাং $x_j^{(i)}$ দ্বারা বোঝানো হচ্ছে i -তম ট্রেনিং ডেটার j -তম ফিচার। আরো ভেঙে বলি, টেবিল 3.5.1 থেকে দেখুন, এখানে $x^{(1)}$ মানে হচ্ছে টেবিলের প্রথম রো, যেটি প্রথম ট্রেনিং ডেটা নির্দেশ করছে। একইভাবে $x^{(2)}$ মানে টেবিলের দ্বিতীয় রো, যেটি দ্বিতীয় ট্রেনিং ডেটা নির্দেশ করছে।

আর $x_1^{(1)}$ মানে হচ্ছে টেবিলের প্রথম ট্রেনিং ডেটার প্রথম ফিচারের মান, অর্থাৎ প্রথম রো-এর প্রথম কলাম-এর মান অর্থাৎ 2। একইভাবে $x_2^{(2)}$ মানে হচ্ছে দ্বিতীয় ট্রেনিং ডেটার দ্বিতীয় ফিচারের তালু অর্থাৎ দ্বিতীয় রো-এর দ্বিতীয় কলাম-এর ভ্যালু অর্থাৎ 32।

এখন দেখুন, $j = 0$ এর জন্য,

$$\frac{\partial}{\partial w_0} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_0^{(i)}$$

আমরা হাইপোথিসিস দাঁড় করানোর সময় প্রথমেই ধরে নিয়েছিলাম, $x_0 = 1$

সুতরাং

$$\frac{\partial}{\partial w_0} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot 1$$

অর্থাৎ, $j = 0$ -এর জন্য

$$\frac{\partial}{\partial w_0} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}$$

আর বাকি সব $j = 1, 2, 3, \dots, n$ -এর জন্য,

$$\frac{\partial}{\partial w_j} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_j^{(i)}$$

অর্থাৎ, $j = 1$ -এর জন্য,

$$\frac{\partial}{\partial w_1} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_1^{(i)}$$

আবার $j = 2$ -এর জন্য

$$\frac{\partial}{\partial w_2} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_2^{(i)}$$

এভাবে আমরা যতগুলো ফিচার নেব, প্রতিটির সাপেক্ষে একবার করে $J(w_0, w_1, w_2, \dots, w_n)$ -কে আমাদের পার্শিয়াল ডিফারেন্সিয়েশন করতে হবে এবং সেই অনুযায়ী প্যারামিটার আপডেট করতে হবে।

তাহলে এই হচ্ছে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ব্যবহার করে কীভাবে আমরা মাল্টিভ্যারিয়েট লিনিয়ার রিপ্রেশন করতে পারি তার বিবরণ।

পরিচ্ছেদ ৩.৬ : ভেস্ট্র অ্যালজেব্রা (Vector Algebra) ব্যবহার করে লিনিয়ার রিপ্রেশন

এটি লিনিয়ার রিপ্রেশন বের করার সর্বশেষ পদ্ধতি যেটি আমরা দেখব। এটি সরাসরি গাণিতিক একটি পদ্ধতি, এখানে গ্রেডিয়েন্ট ডিসেন্টের মতো প্যারামিটার আপডেট করার কোনো ব্যাপার নেই। এটি বরং আমাদের প্রথম ব্যবহৃত নিউমেরিকাল অ্যানালাইসিস পদ্ধতিটির মতো একটি সরাসরি পদ্ধতি, যেখানে লিনিয়ার অ্যালজেব্রা ব্যবহৃত হয়েছে। পরিচ্ছেদ ৩.২-এ ব্যবহৃত সরাসরি পদ্ধতির সঙ্গে এই পদ্ধতির পার্থক্য হচ্ছে, পরিচ্ছেদ ৩.২-এর পদ্ধতিটি ব্যবহার করা যাব শুধু ইউনিভ্যারিয়েট লিনিয়ার রিপ্রেশনের ক্ষেত্রে, কারণ সেখানে শুধু একটি ফিচার বা ভ্যারিয়েবল নিয়ে কাজ করা হয়। কিন্তু ভেস্ট্র অ্যালজেব্রা ব্যবহার করে আমরা একাধিক ফিচারের জন্য অর্থাৎ মাল্টিভ্যারিয়েট লিনিয়ার রিপ্রেশন-এর জন্য সরাসরি গাণিতিক পদ্ধতি ব্যবহার করে আমাদের সব প্যারামিটারের মান একেবারে বের করতে পারব।

আমরা আগের পরিচ্ছেদে ইতিমধ্যেই দেখেছি, মাল্টিভ্যারিয়েট লিনিয়ার রিপ্রেশনের ক্ষেত্রে আমাদের হাইপোথিসিস ছিল,

$$h_w(x) = w^T x$$

এবং আমাদের কস্ট ফাংশন ছিল,

$$J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

এখন আমরা যদি আমাদের সব ট্রেনিং ডেটাকে একটি ম্যাট্রিক্সের মধ্যে নিই, যার প্রতিটি রো হচ্ছে একটি করে ট্রেনিং ডেটা আর প্রতিটি কলাম হচ্ছে একেকটি ফিচার, তাহলে ম্যাট্রিক্সটি হবে $m \times (n + 1)$ ডাইমেনশনের একটি ম্যাট্রিক্স, যেখানে মোট ট্রেনিং ডেটার সংখ্যা m , মোট ফিচারের সংখ্যা n ,

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{bmatrix}$$

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

সেই সঙ্গে, যদি আমরা আরো একটি ম্যাট্রিক্স নিই, যেটি হবে একটি কলাম ম্যাট্রিক্স এবং তাতে m -সংখ্যক সারি থাকবে, যা আমাদের m -সংখ্যক দেনিং ডেটার প্রতিটির আউটপুটের আসল মানগুলো ধারণ করবে, তাহলে ম্যাট্রিক্সটি হবে $m \times 1$ ডাইমেনশনের একটি ম্যাট্রিক্স,

$$Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

সবশেষে, আমাদের n -সংখ্যক ফিচারের জন্য সমস্ত ওয়েইট ($w_0, w_1, w_2, \dots, w_n$) নিয়ে একটি ওয়েইট ভেট্রে যদি চিন্তা করি, সেটি হবে $(n + 1) \times 1$ ডাইমেনশনের একটি ম্যাট্রিক্স,

$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

এখন আমরা সরাসরি নিচের সূত্র প্রয়োগ করে প্রতিটি ওয়েইট ($w_0, w_1, w_2, \dots, w_n$)-এর অপটিমাল মান একবারেই পেয়ে যেতে পারি,

$$\underline{w = (x^T x)^{-1} x^T y}$$

এখন থেকে w -এর মান পেয়ে গেলে সেটি আমাদের হাইপোথিসিস $h_w(x) = w^T x$ -এ বসিয়ে দিলেই আমরা পেয়ে যাব আমাদের কাঙ্ক্ষিত হাইপোথিসিসের সর্বশেষ সমীকরণ। এটিই হচ্ছে ভেট্রে অ্যালজেব্রা ব্যবহার করে এক বা একাধিক ফিচারের জন্য লিনিয়ার রিগ্রেশন করার পদ্ধতি।

এখন প্রশ্ন হচ্ছে কখন আমরা গ্রেডিয়েন্ট ডিসেন্ট ব্যবহার করব, আর কখন আমরা ভেট্রে অ্যালজেব্রা ব্যবহার করব? এর উত্তর হচ্ছে, যদি ফিচার সংখ্যা $n \leq 10,000$ হয়, তাহলে আমরা ভেট্রে অ্যালজেব্রা ব্যবহার করতে পারি, কেননা এতে কোনো লুপ বা ইটারেশন নেই এবং কোনো শুধু নেওয়ার ব্যাপার নেই। তাই খুব দ্রুতই আমরা সমাধানে পৌঁছে যেতে পারি।

শুধু $n \leq 10,000$ -এর জন্যই কেন আমরা এটি ব্যবহার করতে পারি, এর পেছনে একটি কারণ হচ্ছে, w -এর সমীকরণটির দিকে তাকালে বুবাবেন, এখানে তিনটি ম্যাট্রিক্স মাল্টিপ্লিকেশন (Matrix Multiplication) অপারেশন করা হয়েছে এবং একটি ম্যাট্রিক্স ইনভার্শন (Matrix Inversion) অপারেশন করা হয়েছে। ধরে নিচ্ছি, সবাই-ই ম্যাট্রিক্স গুণন কীভাবে করতে হয় জানেন। আমি এখানে শুধু একটি 2×2 ম্যাট্রিক্স ইনভার্শনের সূত্রটি দিচ্ছি।

$$\text{ধরি, } A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\text{তাহলে, } A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

আশা করি দেখেই বুঝতে পারছেন, এটি বেশ জবরজৎ একটি পদ্ধতি এবং যদি আমাদের n -এর মান অনেক বড়ো হয় ($10,000$ -এর চেয়েও বড়ো, ধরি $11,000$) তাহলে আমাদের –

X হবে $11,000 \times 11,000$ এই ডাইমেনশনের একটি ম্যাট্রিক্স।

X^T -ও হবে $11,000 \times 11,000$ এই ডাইমেনশনের একটি ম্যাট্রিক্স।

এখন এত বড়ো ডাইমেনশনের দুটি ম্যাট্রিক্সের গুণফল অবশ্যই Computationally Expensive হবে। এখানেই শেষ নয়। এই গুণফল $X^T X$ -এর ইনভার্স নিতে হবে, যেটি ম্যাট্রিক্স গুণফলের চেয়েও অনেক বেশি Computationally Expensive। সবশেষে, এই গুণফলের সঙ্গে আবার X^T গুণ করে তার সঙ্গে Y ম্যাট্রিক্সটি গুণ করলে আমরা সর্বশেষ ফলাফল পাব। এত বড়ো Computation কম্পিউটারের পক্ষে করা অনেক সময়সাপেক্ষ একটি ব্যাপার, তাই $n \leq 10,000$ পর্যন্ত আমরা এই ভেষ্টর অ্যালজেব্রা মেথড ব্যবহার করব।

কিন্তু যদি ফিচার সংখ্যা $n > 10,000$ হয়, তাহলে গ্রেডিয়েন্ট ডিসেন্ট ব্যবহার করতে হবে। কেননা, গ্রেডিয়েন্ট ডিসেন্ট ফিচার সংখ্যা n -এর মান অনেক বড়ো হলেও ভালোভাবেই সমাধানে পৌঁছুতে পারে, যদিও এতে ইটারেশন অপারেশন আছে এবং সঠিকভাবে α বেছে নেওয়ার ব্যাপার আছে, যার কারণে এটি প্রায়ই ভেষ্টর অ্যালজেব্রা মেথডের চেয়ে অনেক বেশি সময় ধরে চলে।

তাহলে এই হলো ভেষ্টর অ্যালজেব্রা ব্যবহার করে কীভাবে আমরা লিনিয়ার রিগ্রেশন করতে পারি তার বিস্তারিত বর্ণনা।

পরিচ্ছেদ ৩.৭ : বায়াস (Bias), ভ্যারিয়েন্স (Variance) ও রেগুলারাইজেশন (Regularization)

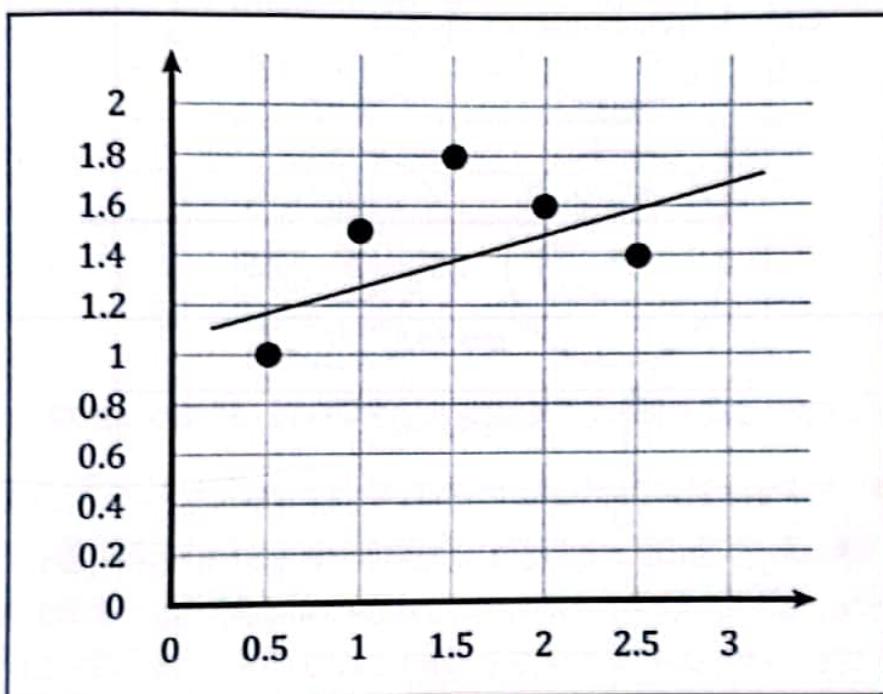
পরবর্তী সময়ে আরো কতগুলো ধারণা আমাদের প্রয়োজন হবে। সেগুলো আমি এখানেই আলোচনা করছি।

প্রথমেই আমরা বুঝে নিই ওভারফিটিং (Overfitting) ও আন্ডারফিটিং (Underfitting) কী। নাম শুনেই বোঝা যাচ্ছে যে, ওভারফিটিং মানে হচ্ছে বেশি হস্তিত্ব করে বেশি বেশি ফিট হয়ে গেছে, আবার আন্ডারফিটিং মানে হচ্ছে একেবারেই ফিট হয় নাই।

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

আপনি বিষয়টিকে এভাবে চিন্তা করতে পারেন – ধরে নিন আপনি একটি শার্ট কিনেছেন। সেই শার্ট ওভারফিটিং হওয়া মানে হচ্ছে সেটি গায়ের সঙ্গে, শরীরের প্রতিটি ভাঁজের সঙ্গে একেবারে ঔটোসৌটোভাবে লেগে আছে। আপনি যদি, সামান্য একটু মোটা হয়ে যান, সেই শার্ট আর আপনার গায়ে খাটবে না। একইভাবে আন্ডারফিটিং মানে হচ্ছে, আপনি এমন একটি শার্ট কিনেছেন যেটি আপনার গায়ে একেবারেই ফিটিং হয়নি, একদম ঢেলাঢালা হয়ে আছে।

নিচের গ্রাফটি দেখুন (গ্রাফ 3.7.1)। এই গ্রাফটিতে, আমরা যে সরলরেখা হাইপোথিসিস হিসেবে নিয়েছি, সেটি আমাদের ট্রেনিং ডেটায় একেবারেই ফিট করছে না, প্রতুর এরর বা কষ্ট আছে। এই ধরনের ক্ষেত্রে বলা হয় যে, আমাদের হাইপোথিসিস আন্ডারফিট হয়েছে, অর্থাৎ একেবারেই ফিটিং হ্যানি। ডেটা আছে একরকম, আর হাইপোথিসিস হয়েছে অন্যরকম।

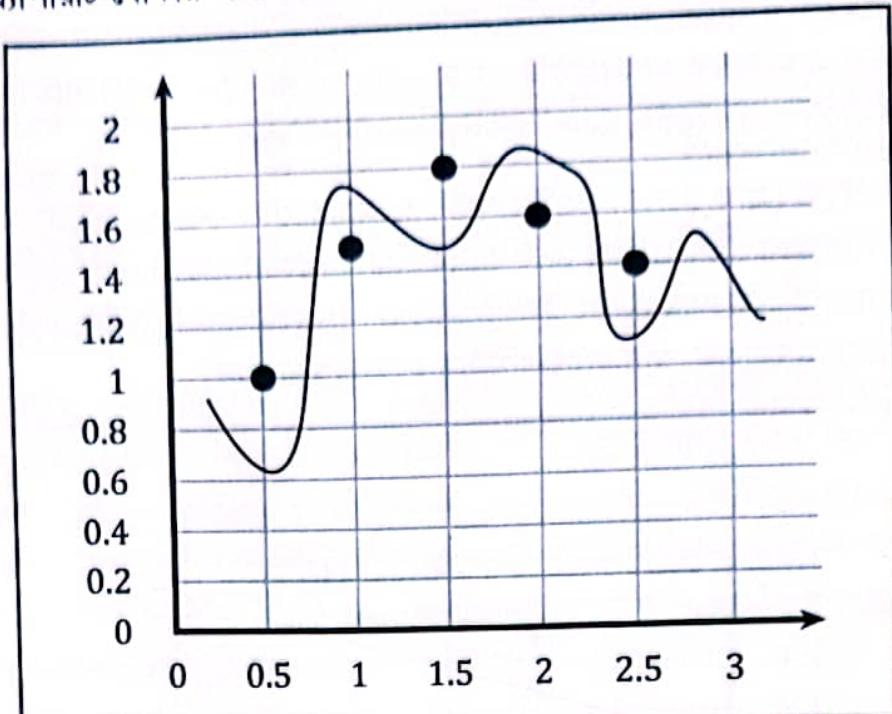


গ্রাফ 3.7.1

এই ধরনের আন্ডারফিট হাইপোথিসিসের ক্ষেত্রে বলা হয় যে এদের হাই বায়াস (High Bias) আছে, অর্থাৎ, হাইপোথিসিসগুলো আগে থেকেই ধারণা করে নিয়েছে যে ডেটা কীরকম হবে (আমাদের এই ক্ষেত্রে সরলরেখিক) এবং সেই ধারণা ট্রেনিং ডেটা পর্যবেক্ষণ করার পরেও পরিবর্তিত হবে না। হাইপোথিসিস তাঁর নিজের ধারণার প্রতিই পক্ষপাতী থাকে। এটি অনেকটা জেন করে গোঁ ধরে বসে থাকা ছোটো বাচ্চাদের মতো, যতই বোঝানো হোক (ট্রেনিং দেওয়া হোক) সে কথা শুনবে না (নিজের আকৃতি বদলাবে না)।

একইভাবে, ওভারফিটিং বলতে বোঝায় যে, আমাদের হাইপোথিসিস খুবই জটিল, প্যাঁচানো একটি হাইপোথিসিস এবং সেটি শুধু ট্রেনিং ডেটাতেই খুব ভালোভাবে ফিট করে। ট্রেনিং ডেটার বাইরে

যখন অন্য নতুন কোনো ডেটা আসে, সেই ডেটাকে আমাদের এই হাইপোথিসিস ফিট করাতে পারবে না। ব্যাপারটি বোঝার জন্য নিচের গ্রাফ দেখুন (গ্রাফ 3.7.2)।



গ্রাফ 3.7.2

গ্রাফে হাইপোথিসিসের যেই কার্ড আঁকা হয়েছে, তার সমীকরণ ধরা যাক এরকম –

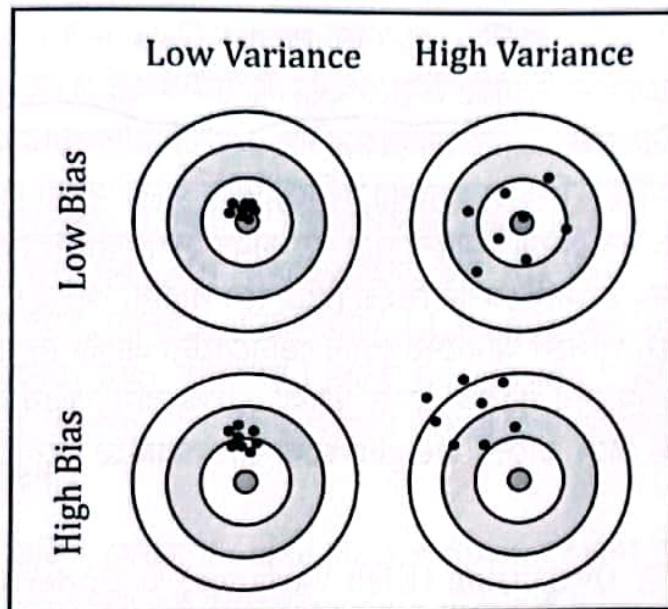
$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_3x_1^2 + w_4x_2^3$$

এখন দেখুন, কী বিদঘৃটে একটি হাই-অর্ডার পলিনোমিয়াল হাইপোথিসিস (প্রধান চলক x -এর সর্বোচ্চ ঘাত/পাওয়ার 1-এর বেশি), তাই না? এত কঠিন, ঘোরানো-প্যাঁচানো হাইপোথিসিস কেন ব্যবহার করা হয়েছে – শুধু যেন তা আমাদের ট্রেনিং ডেটাকে ঠিকভাবে ফিট করে, একটি ট্রেনিং পয়েন্টও যেন বাদ না যায়, অর্থাৎ, ট্রেনিং ডেটার সাপেক্ষে হাইপোথিসিসের কস্ট যেন শূন্য হয়।

এই ধরনের হাইপোথিসিস ট্রেনিং ডেটাতে খুব ভালোভাবে ফিট করবে অবশ্যই, কিন্তু সমস্যা হবে যখন ট্রেনিং ডেটার বাইরে অন্য কোনো ডেটা আসবে। তখন আর আমাদের এই প্যাঁচানো হাইপোথিসিস ভালোভাবে কাজ করবে না। এই ধরনের হাইপোথিসিস আসলে ট্রেনিং ডেটাতে এত ভালোভাবে ওভারফিট করে যে, এর বাইরে অন্য কোনো ডেটার জন্য সে নিজেকে জেনারালাইজ করতে পারে না আর; একেই বলে ওভারফিটিং। বলা হয় যে, Overfitted হাইপোথিসিসের High Variance থাকে। একটি হাইপোথিসিস দ্বারা ডিফাইন করা মডেল আসলে যত বেশি জটিল হয়, তত তার High Variance থাকে এবং তত তার Overfitted হওয়ার সম্ভাবনা থাকে।

অধ্যায় ৩ : লিনিয়ার রিজেশন (Linear Regression)

Bias-Variance Tradeoff বলে একটি টার্ম আছে, যেটি আপনারা কোথাও কোথাও হয়তো দেখবেন। আমাদের লক্ষ্য সব সময় থাকে আমরা Low Bias, Low Variance – এরকম কোনো একটি মডেল দাঁড় করাব। কিন্তু, বেশিরভাগ সময়েই আমরা এমন মডেল দাঁড় করাই, যেটাতে High Variance, High Bias কিংবা, High Variance, Low Bias অথবা Low Variance, High Bias থাকে।



ছবি 3.7.1

ওপরের ছবিটি (ছবি 3.7.1) দেখলে হয়তো আপনাদের আরেকটু পরিষ্কার ধারণা হবে ব্যাপারটা সম্পর্কে। অনেকেই হয়তো ডার্ট ছোড়ার খেলাটা খেলেছেন। আপনাদের হাতে কতগুলো ডার্ট থাকে, সেটি দিয়ে মাঝখানের ওই লাল বৃত্তটির মধ্যে ছুড়ে বিধাতে হয়। আমাদের এই ছবি 3.7.1-এ চারটি কেস দেওয়া আছে দেখবেন।

আমাদের লক্ষ্য থাকে সব সময় Low Bias (Accuracy), Low Variance (Consistency) পাওয়া, অর্থাৎ আমি যে ডার্ট ছুড়ব সেটি যেন মাঝখানের ওই লাল ছোটো বৃত্তে বেঁধে (Accuracy) এবং আমি যদি পরপর 10টি ডার্ট ছুড়ি, তাহলে 10টিই যেন লাল বৃত্তে গিয়ে বেঁধে (Consistency)। কিন্তু বেশিরভাগ সময়েই আমাদের হয়তো এটি achieve করা সম্ভব হয় না, প্রায় অসম্ভবই বলা চলে ব্যাপারটা। হয়তো, Accuracy ভালো হলে মডেল Inconsistent হয়, আবার Consistency থাকলেও মডেল Accurate হয় না। আবার কখনো কখনো কোনোটাই থাকে না।

এর কারণ হচ্ছে, সহজ করে বললে, একটি মডেলের Low Variance থাকা মানে মডেলটা Less Complex। সুতরাং, আমাদেরকে Low Variance যদি achieve করতে হয়, তাহলে Less

Complex কোনো একটি মডেল দিয়ে কাজ করতে হবে। আবার যদি আমরা Low Bias চাই, তাহলে আমাদেরকে Highly Complex মডেল ব্যবহার করতে হবে। এখন দেখুন, আমরা যদি দুটোই একসঙ্গে অর্জন করতে চাই, তাহলে আমাদের এমন একটি মডেল বেছে নিতে হবে, যেটি একই সঙ্গে Highly Complex আবার খুবই Low Complex, যেটি কখনোই সন্তুষ্ট নয়, তাই না? তাই আসলে আমাদেরকে মাঝামাঝি কোনো একটি পয়েন্ট খুঁজে বের করতে হয়, এটাই Bias-Variance Tradeoff।

Overfitting প্রধানত হয় যখন ফিচার ডেটার সংখ্যা আমাদের ট্রেনিং ডেটার চেয়ে অনেক বেশি হয়ে যাবে তখন (ডেটাসেটের টেবিলে কলামের সংখ্যা, সারির সংখ্যার চেয়ে বেশি হয়)। এর কমানোর দুটো উপায় আছে – ফিচারের সংখ্যা কমিয়ে নিয়ে আসা অথবা রেগুলারাইজেশন করা। ফিচারের সংখ্যা আসলে সব ক্ষেত্রে কমানো যায় না, কারণ দেখা যায় সবগুলো ফিচারই কোনো-কোনোভাবে আমাদের ক্লাসিফিকেশন কিংবা রিপ্রেশনে সাহায্য করছে। সুতরাং, একটি ফিচার ভ্যালু কমিয়ে ফেলা মানে আসলে আমাদের গোটা ডেটাসেটের একটি বিশাল পরিমাণ বাদ দিয়ে দেওয়া, যেটি আসলে আমাদের পরবর্তী সময়ে ভালো একিউরেসি পাওয়ার অন্তরক হতে পারে। তাই, সবগুলো ফিচারই রেখে তাদের weight-গুলো রেগুলারাইজ করে নেওয়াটাই বেশিরভাগ ক্ষেত্রে উত্তম পদ্ধা।

এতক্ষণ আমরা দেখলাম Overfitting (High Variance) ও Underfitting (High Bias) বলতে আসলে কী বোঝায়। এখন আসি রেগুলারাইজেশনে।

রেগুলারাইজেশন মানে হচ্ছে সোজা বাংলায় হচ্ছে কোনো কিছুর গুরুত্বকে বা প্রভাবকে অন্য কোনো প্রভাবকের সাহায্যে কমানো-বাড়ানো। একে ফ্যানের রেগুলেটরের সঙ্গে তুলনা করা যেতে পারে। আমরা যখন ফ্যানের স্পিড কমানোর চেষ্টা করি, তখন আমরা কী করি? আমরা কি ফ্যানের পাখা ধরে বুলে পড়ি স্পিড কমানোর জন্য? কিংবা যদি আমরা ফ্যানের স্পিড বাড়াতে চাই তাহলে কী করি? ফ্যানের পাখা ধরে বাই বাই করে ঘোরানো শুরু করি কি? মোটেও না? আমরা ব্যবহার করি রেগুলেটর নামে ছোট্ট একটি যন্ত্র, যেটি ঘুরিয়ে ঘুরিয়ে আমরা ফ্যানের স্পিড কমাই-বাড়াই।

রেগুলারাইজেশন ভালো কাজ করে সেইসব ক্ষেত্রে, যখন আমাদের বেশ অনেকসংখ্যক ফিচার ডেটা আছে এবং কোনো ফিচার ডেটাই আমাদের বাদ দেওয়ার উপায় নেই, সেই সব ক্ষেত্রে প্রতিটি ফিচারের সঙ্গে Associated থেকে Weight-গুলোকে পরিবর্তিত করে এইসব ফিচার ভ্যালুগুলোর গুরুত্ব কমানো-বাড়ানো যায়।

যেরকম, একটি উদাহরণ দিই। ধরা যাক, নিচের সমীকরণটি –

$$A \cdot B = 1$$

এই সমীকরণে, যদি, A ও B দুটিকেই 1 ধরে নিই, তাহলে সমীকরণের উভয় পক্ষ সমান থাকে। এখন যদি আমি চাই, আমি এই সমীকরণে B-এর প্রভাব কমাব, তাহলে কী করব?

একটু খেয়াল করে দেখুন, আমরা যদি আন্তে আন্তে A-এর মান বাড়াতে থাকি 1 থেকে 10, 100, 1000, 10000, 100000 এভাবে, তাহলে সমীকরণের উভয় পাশ সমান রাখার জন্য B-এর মান কিন্তু কমতে থাকবে 1 থেকে 0.1, 0.01, 0.001, 0.0001, 0.00001 এইভাবে। এই ঘটনাটি রেগুলারাইজেশন বোঝার জন্য একটি সাধারণ উদাহরণ। এটি দেওয়া হলো শুধু ব্যাপারটি সম্পর্কে একটু ধারণা পাওয়ার জন্য। আরেকটি জিনিস, যেহেতু B-কে কমানো-বাড়ানো আমরা A-এর মান বাড়ানো-কমানোর মাধ্যমে করছি, তাই A-কে বলা হয় Regularization Parameter।

আমরা রেগুলারাইজেশন মূলত ব্যবহার করি ওভারফিটিং কমানোর জন্য। আর ওভারফিটিং মূলত হয় যদি আমাদের মডেল খুব বেশি কমপ্লেক্স হয়, তাই না? আমরা মোটামুটি দুই ধরনের রেগুলারাইজেশনের সম্পর্কে জানব – একটি হচ্ছে L1 Regularization, আরেকটি হচ্ছে L2 Regularization। এদের আবার যথাক্রমে LASSO Regression ও RIDGE Regression-ও বলা হয়। কেন বলা হয়, এগুলো আপাতত আমাদের বইয়ের একত্তিয়ারের বাইরে, এগুলো আরো কিছুটা অ্যাডভান্সড ধারণা।

এ দুটোর ধারণা বেশ জটিল, সেজন্য Sparse Matrix, Sparsity এবং আরো কিছু ধারণা জানা প্রয়োজন যেটি আসলে আপাতত এই হাতেখড়ির সময়ে না জানলেও চলবে বলে আমি মনে করি। সুতরাং, আপাতত সেগুলো বাদ রাখছি। আমাদের শুধু আপাতত এতটুকু জানলেই চলবে যে, যে-কোনো ধরনের রেগুলারাইজেশনের ক্ষেত্রেই আমরা আমাদের Weight Vector-এর মানগুলোকে কমিয়ে আনার চেষ্টা করি যতটুকু পারা যায়। এর একটি গালভরা নাম আছে – Penalizing the Weight Vectors।

L1 Regularization-এর ক্ষেত্রে আমরা যেটি রেগুলারাইজেশনের জন্য ব্যবহার করি, তা হলো –

$$\frac{\lambda}{2m} \sum_{j=1}^n |w_j|$$

আর L2 Regularization-এর ক্ষেত্রে আমরা যেটি ব্যবহার করি সেটি হলো –

$$\frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

উভয় ক্ষেত্রেই ১ হচ্ছে রেগুলারাইজেশন প্যারামিটার, যেটি কিনা Weight Vector কমানোর সঙ্গে সমানুপাতিক। λ -এর মান যত হবে, যে-কোনো মডেলের তত বেশি কমপ্লেক্স থেকে সহজ হবার প্রবণতা বাড়বে। λ -এর মান শূন্য হলে কোনো রেগুলারাইজেশন হবে না।

মেশিন লার্নিং অ্যালগরিদম

একটি ছোটো উদাহরণ দিই। একটু আগেই আমরা ওভারফিটিং পড়ার সময় একটি বিচ্ছিন্ন, প্যাঁচানো ও জটিল হাইপোথিসিস দেখলাম, যেটি ছিল এরকম –

$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_3x_1^2 + w_4x_2^3$$

এখন, দেখে কিছুটা হয়তো আপনারা আইডিয়া করতে পারছেন, যে এই হাইপোথিসিসে ওভারফিটিং হওয়ার সন্তান বেশ ভালো।

এখন, যদি আমরা ওভারফিটিং থেকে বাঁচতে চাই, তাহলে আমাদের একমাত্র ভরসা রেগুলারাইজেশন।

যেহেতু আমরা লিনিয়ার রিগ্রেশন করছি, এর জন্য কস্ট ফাংশন কীরকম হবে সেটি আমরা আগেই দেখেছি –

$$\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

এখন, রেগুলারাইজেশন করা জন্য আমরা যেটি করব যে, আমাদের কস্ট ফাংশনের সঙ্গে একটি রেগুলারাইজেশন টার্ম যুক্ত করে দেব ঠিক এভাবে –

$$\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

এখন আমরা জানি যে, আমাদের অবজেকটিভ ফাংশন হচ্ছে এই কস্ট ফাংশনটিকে minimize করা। লক্ষ করুন, আমাদের ওপরের কস্ট ফাংশনটিতে দুটি টার্ম আছে, একটি সাধারণ Squared Error টার্ম, আরেকট হচ্ছে রেগুলারাইজেশন টার্ম।

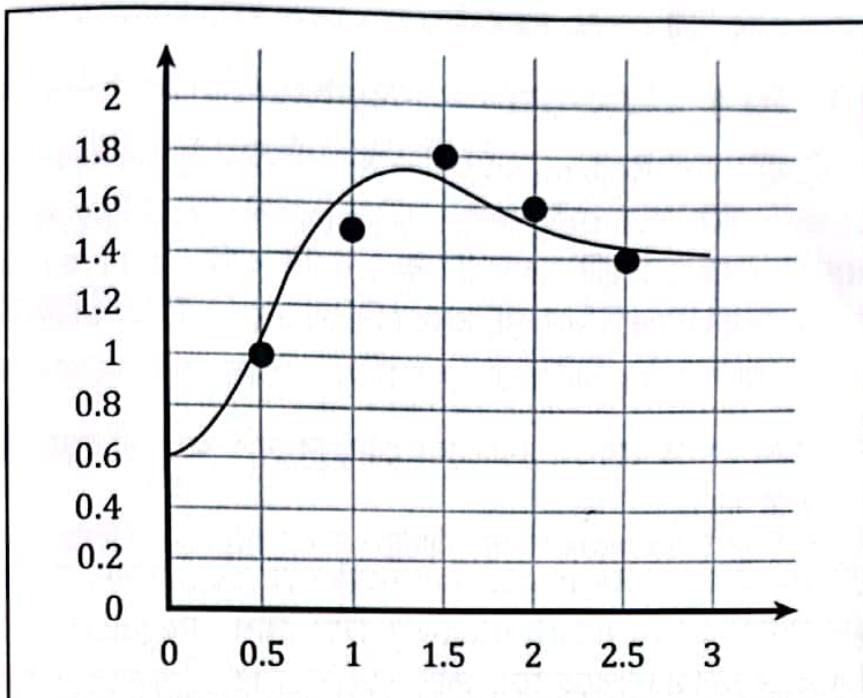
এখন, একটু ভালোমতো চিন্তা করে দেখুন তো, যেহেতু আমাদের লক্ষ্য হচ্ছে কস্ট ফাংশন মিনিমাইজ করা এবং আমাদের প্রথম টার্ম সাধারণভাবেই মিনিমাইজ হচ্ছে আগের মতোই সেটাতে কোনো সমস্যা নেই; কিন্তু যদি আমরা রেগুলারাইজেশন টার্মে λ -এর বেশ বড়োসড়ে একটি মান দিয়ে দিই (ধৰা যাক, $\lambda = 1000$), তাহলে, সেই বড়ো মানের প্রভাব কমানোর জন্য (যেহেতু আমরা মিনিমাইজ করছি) $\sum_{j=1}^n w_j^2$ -এর মানকে খুব ছোটো মানের দিকে (প্রায় শূন্যের কাছাকাছি) কমে আসতে হবে, তাই না?

অর্থাৎ λ -এর বড়ো মানের প্রভাব কস্ট ফাংশনে কমিয়ে আনার জন্য আমাদের Weight Vector-এর প্রতিটি weight কিংবা parameter-কে একেবারে কমিয়ে শূন্যের কাছাকাছি চলে আসতে হবে।

এটিই হচ্ছে রেগুলারাইজেশনের ধারণা এবং একটু চিন্তা করে দেখুন, আমাদের weight vector-গুলোর মান যত ছোটো হতে থাকবে, তত আমাদের সেই বিচ্ছিন্ন হাইপোথিসিসটি এরকম

অধ্যায় ৩ : লিনিয়ার রিট্রেশন (Linear Regression)

আঁকাৰ্বঁকা কাৰ্ড আকাৰ থেকে কিছুটা সুন্দৱ হয়ে গ্ৰাফ 3.7.3-এৰ মতো একটি আকাৰ হয়তো (কাছাকাছি কিছু একটা) ধাৰণ কৰবে।



গ্ৰাফ 3.7.3

গ্ৰাফটা আগেৰ মতো আঁকাৰ্বঁকা থেকে এৱকম সুন্দৱ smooth হয়ে আসাৰ মানে হচ্ছে আমাদেৱ মডেলেৰ complexity আগেৰ চেয়ে অনেকাংশেই কমে এসেছে। অৰ্থাৎ, আমৱা আমাদেৱ মডেলে ওভাৱফিটিং হওয়াৰ সম্ভাৱনা অনেকটাই কমিয়ে এনেছি।

এখানে লক্ষণীয় যে, আমৱা কিন্তু সৱাসৱি weight vector-গুলোৰ মান কমাইনি, বৰং λ -এৰ মান এমনভাৱে নিয়েছি, যাতে weight vector-গুলো কমে আসতে বাধ্য হয়, অৰ্থাৎ একটিৰ মান বাড়িয়ে আমৱা আৱেকটিৰ মান কমিয়েছি, যেটি ছিল আমাদেৱ রেগুলাৱাইজেশনেৰ মূল ধাৰণা।

এখানে আৱেকটি ব্যাপার একটু খেয়াল রাখতে হবে, আমৱা রেগুলাৱাইজেশন কৰাৰ সময় রেগুলাৱাইজেশন টাৰ্ম হিসেবে ব্যবহাৰ কৰছি—

$$\frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

এখন, একটু লক্ষ কৰবেন যে আমৱা কিন্তু এখানে $j = 1$ থেকে n পৰ্যন্ত weight vector-এৰ মান নিয়েছি, একেবাৱে প্ৰথম যে weight vector ছিল আমাদেৱ w_0 , তাকে কিন্তু আমৱা রেগুলাৱাইজ কৰছি না।

মেশিন লার্নিং অ্যালগরিদম

এখন, যদি আমরা λ -এর মান খুব বেশি বড়ো নিয়ে ফেলি, তাহলে, w_1 থেকে w_n -এর পর্যন্ত সবগুলোর মান রেণুলারাইজ করে আমরা শূন্যের একদম কাছাকাছি নিয়ে এলেও w_0 -এর মান কিন্তু যা ছিল, তা-ই থেকে যাবে।

$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_3x_1^2 + w_4x_2^3$$

তখন ওপরের এই হাইপোথিসিসের, w_0 বাদে বাকি সবগুলো টার্ম শূন্যের কাছাকাছি চলে যাবে, ফলে – $h_w(x) \approx w_0$ হয়ে যাবে, যেটি মূলত একটি সরলরেখা। অনেকটা $y = 4$, এই ধরনের x -অক্ষের সমান্তরাল কোনো একটি সরলরেখার কথা বলছি আমি। এখন শুধু এরকম একটি সরলরেখা যদি আমাদের হাইপোথিসিস হয়, তাহলে নিশ্চিতভাবে সেটি আন্ডারফিটিং হয়ে যাবে, তাই না?

তাই, আমাদের খেয়াল রাখতে হবে যে, λ -এর মান যেন খুব বেশি বড়ো না হয়ে যায়; আবার খুব বেশি ছোটোও না হয়ে যায়।

আমরা মোটামুটি এই বইতে L2 Regularization নিয়েই কাজ করব, কেননা সেটির ওভারফিটিং এড়িয়ে যাওয়ার ক্ষমতা L1 Regularization-এর চেয়ে বেশি। Regularization আমাদের অ্যালগরিদমে কীভাবে ব্যবহার করতে হবে, সেটি নিয়ে বিস্তারিত লেখা হবে পরিচ্ছেদ 8.8-এ। সেটি পড়লে আশা করি রেণুলারাইজেশনের ব্যবহার সম্পর্কে ধারণা পরিষ্কার হবে আরো।

পরিচ্ছেদ 3.8 : গ্রেডিয়েন্ট ডিসেন্ট পদ্ধতির সাহায্যে লিনিয়ার রিগ্রেশনের উদাহরণ

আশা করি, ইতিমধ্যে আমাদের লিনিয়ার রিগ্রেশনের ওপরে বেশ ভালো একটি ধারণা হয়েছে। এখন, গ্রেডিয়েন্ট ডিসেন্ট ব্যবহার করে কীভাবে এই লিনিয়ার রিগ্রেশন করা যায়, সেটি একটু দেখা যাক।

X	Y
1	1
2	3
4	3
3	2
5	5

টেবিল 3.8.1

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

টেবিল 3.8.1-এ দেখুন আমাদের পাঁচটি ডেটা পয়েন্টের মান দেওয়া আছে। এটিই আমাদের লিনিয়ার রিগ্রেশনের জন্য ট্রেনিং ডেটা হবে। প্রথমে আমরা এই ট্রেনিং ডেটা দিয়ে আমাদের লিনিয়ার রিগ্রেশন মডেলকে ট্রেইন করব এবং সেটি করব প্রেডিয়েন্ট ডিসেন্ট ব্যবহার করে।

আমরা এই উদাহরণটি চেষ্টা করব একেবারে সহজ করে বুঝতে। যাঁদের এটি ধরতে সমস্যা হবে, তাঁরা আরেকবার পরিচেদ ৩.৩ ও ৩.৪ পড়ে নেবেন।

প্রথমেই, আমাদের লিনিয়ার রিগ্রেশনের হাইপোথিসিস হবে,

$$h_w(x) = w_0 + w_1 x$$

আর এটি ছিল আমাদের কস্ট ফাংশন, যেটিকে আমাদের মিনিমাইজ করতে হবে,

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

এবং আমাদের প্রতিটি Weight অর্থাৎ প্যারামিটার (Parameter) আপডেট হবে এই সূত্র অনুসারে –

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1)$$

যখন $J = 0$, তখন,

$$w_j = w_j - \alpha \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}$$

যখন $J > 0$, তখন,

$$w_j = w_j - \alpha \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x^{(i)}$$

এখন, শুরুতেই ধরে নিই,

$$w_0 = 0.0$$

$$w_1 = 0.0$$

সুতরাং, আমাদের হাইপোথিসিস দাঁড়ায়,

$$h_w(x) = 0 + 0 \times x$$

এখন তাহলে এই হাইপোথিসিসে আমাদের প্রথম ডেটাপয়েন্ট $(1,1)$ থেকে $x = 1$ যদি ইনপুট দিই, তাহলে মডেলের প্রেডিকশন হবে, $h_w(x) = 0$ ।

মেশিন লার্নিং অ্যালগরিদম

একইভাবে বাকি ডেটাপয়েন্টগুলোর জন্য আমরা যদি হিসাব করি, তাহলে দেখব সবগুলোর জন্যই হাইপোথিসিসের মান আসছে শূন্য। অর্থাৎ মডেল-এর প্রেডিকশন হচ্ছে শূন্য।

নিচের চার্টটি দেখি :

X	Y (Original Output)	$h_w(x)$ (Predicted Output)	Error $= \{h_w(x^{(i)}) - y^{(i)}\}$	$\{h_w(x^{(i)}) - y^{(i)}\}^2$
1	1	0	-1	1
2	3	0	-3	9
4	3	0	-3	9
3	2	0	-2	4
5	5	0	-5	25
Total			-14	48

টেবিল 3.8.2

সুতরাং $w_0 = 0$ ও $w_1 = 0$, এই প্যারামিটারের জন্য আমাদের ডেটাসেটের Cost হবে,

$$\begin{aligned}
 J(w_0, w_1) &= \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 \\
 &= \frac{1}{2 \times 5} (1 + 9 + 9 + 4 + 25) [m = 5, \text{মোট ডেটাপয়েন্টের সংখ্যা}] \\
 &= \frac{48}{10} \\
 &= 4.8
 \end{aligned}$$

4.8 কিন্তু Cost হিসেবে বেশ বড়ো একটি মান, কারণ আমাদের লক্ষ্য ছিল Cost-এর মানকে শূন্যতে নামিয়ে নিয়ে আসব। এখন, আমরা যেহেতু কেবল প্রথম ডেটাপয়েন্ট দেখেছি এতক্ষণে, সুতরাং প্রথম ডেটাপয়েন্ট দেখার পরে আমাদের একবার Weight-এর মান আপডেট করতে হবে।

আমরা ধরে নিই, আমাদের লার্নিং রেট $\alpha = 0.01$

সূত্র অনুসারে,

$$w_0 = 0 - 0.01 \times \frac{1}{5} \times (-14) = 0.028$$

$$w_1 = 0 - 0.01 \times \frac{1}{5} \times (-14) \times 1 = 0.028$$

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

এখন আমরা, এই নতুন $w_0 = 0.028$ ও $w_1 = 0.028$ নিয়ে আমাদের দ্বিতীয় ডেটা পয়েন্ট (2, 3)-এর জন্য হিসাব করব।

$$h_w(x) = 0.028 + 0.028 \times x$$

X	Y (Original Output)	$h_w(x)$ (Predicted Output)	Error $= \{h_w(x^{(i)}) - y^{(i)}\}$	$\{h_w(x^{(i)}) - y^{(i)}\}^2$
1	1	0.056	-0.944	0.891
2	3	0.084	-2.916	8.503
4	3	0.14	-2.86	8.179
3	2	0.112	-1.888	3.564
5	5	0.168	-4.832	23.348
Total			-13.44	44.485

টেবিল 3.8.3

$$\begin{aligned} \text{Cost}, J(w_0, w_1) &= \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 \\ &= \frac{1}{2 \times 5} \times 44.485 \\ &= 4.4485 \end{aligned}$$

দেখা যাচ্ছে যে আমাদের Cost কিন্তু আগের চেয়ে নেমে এসেছে, অর্থাৎ বোঝা যাচ্ছে যে আমাদের প্রেডিয়েন্ট ডিসেন্ট কাজ করছে।

আমরা এভাবে করে বাকি তিনটি ডেটা পয়েন্টের জন্যও হিসাব করব, দেখব আমাদের Cost-এর মান আরো কমবে।

এভাবে আমাদের পাঁচটি ডেটা পয়েন্টের জন্য হিসাবনিকাশ শেষ হলে আমরা বলব যে, আমাদের প্রথম রাউন্ড শেষ হলো, অর্থাৎ প্রথম Epoch শেষ হলো। এরপর আমরা এই পাঁচটি ডেটা পয়েন্ট দিয়ে মডেলকে আরো এক রাউন্ড ট্রেনিং দেব। দ্বিতীয় রাউন্ডের শুরুতে w_0 ও w_1 মান হবে প্রথম রাউন্ড শেষে শেষবারের মতো আপডেট করার পরে w_0 ও w_1 -এর মান যা ছিল সেটি। এভাবে আমরা আরো বেশ কয়েক রাউন্ডে মডেলকে ট্রেইন করব, ততক্ষণ পর্যন্ত যতক্ষণ না আমরা দেখব যে আমাদের Cost-এর মান আর খুব বেশি নামানো যাচ্ছে না।

ধরা যাক, পঞ্চম রাউন্ডের ($5 \times 5 = 25$ বার Weight আপডেট করার পরে) শেষে আমাদের মোট Cost-এর মান দাঁড়িয়েছে 0.35। এর পরের ষষ্ঠ রাউন্ডে (5টি ডেটাপয়েন্টের জন্য আরো 5

বার আপডেট করার পর) সেটি নেমে দাঁড়াল 0.349, তারপরে সপ্তম রাউন্ডের শেষে দাঁড়াল হয়তো 0.345। অর্থাৎ বোঝাই যাচ্ছে, শেষ দুই রাউন্ডে সর্বমোট 10 বার আপডেট করার পরেও আমাদের Cost-এর মান আর খুব একটা কমছে না, এখানেই কস্ট-এর গ্রাফ অনেকখানি ফ্ল্যাট হয়ে গেছে। সুতরাং, আমরা আর যতই ট্রেনিং দিই না কেন, Cost-এর মান আর খুব একটা কমবে না। অর্থাৎ আমাদের ট্রেনিং আমরা এখানে শেষ করতে পারি।

ট্রেনিং শেষ করে, ধরা যাক, আমাদের $w_0 = 0.2308$ ও $w_1 = 0.7904$ ।

তাহলে আমাদের সর্বশেষ হাইপোথিসিস হবে –

$$h_w(x) = 0.2308 + 0.7904 \times x$$

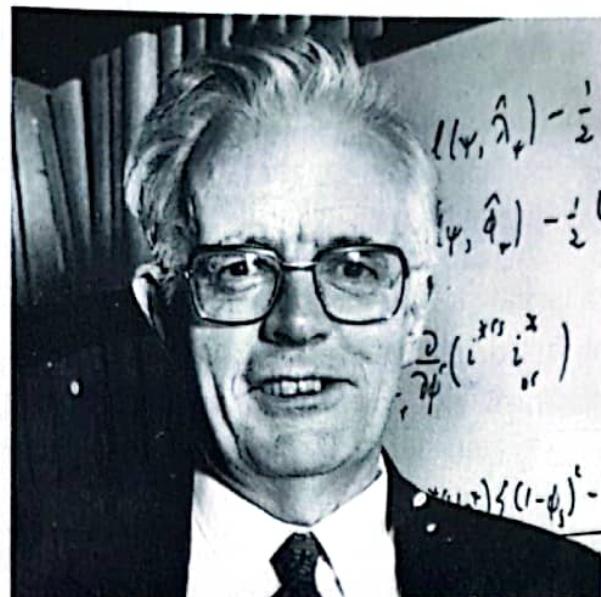
এখন এই হাইপোথিসিসে আমরা x -এর যে-কোনো মান বসালেই প্রেডিকশন পেয়ে যাব যে, y -এর মান কত হবে এবং সেটি প্রকৃত y -এর মানের খুব কাছাকাছিই হবে।

এই ছিল Gradient Descent ব্যবহার করে কীভাবে আমরা লিনিয়ার রিগ্রেশন করতে পারি তার একটি উদাহরণ।

অধ্যায় ৪ : লজিস্টিক রিগ্রেশন (Logistic Regression)

লজিস্টিক রিগ্রেশন (Logistic Regression) পদ্ধতি প্রথমবারের মতো ব্যবহার করেন পরিসংখ্যানবিদ ডেভিড কক্স (David Cox, 1924)। তিনি 1924 সালে বার্মিংহামে জন্মগ্রহণ করেন। তিনি সেইন্ট জনস কলেজ থেকে গণিত নিয়ে পড়াশোনা করেছেন এবং পরবর্তী সময়ে লিডস বিশ্ববিদ্যালয় থেকে পিএইচডি সম্পন্ন করেছেন।

তিনি সারা জীবনে অসংখ্য পুরস্কারে ভূষিত হয়েছেন। এর মধ্যে উল্লেখযোগ্য হলো 'গাই মেডাল' পুরস্কার, যেটি তিনি পেয়েছিলেন রয়্যাল সোসাইটি অব স্ট্যাটিস্টিকস থেকে। সেই সঙ্গে রানি দ্বিতীয় এলিজাবেথ তাঁকে 1985 সালে নাইটহুড প্রদান করেন। তিনি ছিলেন ব্রিটিশ একাডেমির একজন ফেলো।



ছবি 4.1 : ডেভিড কক্স (David Cox, 1924)

পরিচেদ ৪.১ : কন্টিনিউয়াস ও ডিসক্রিট ডেটা (Continuous and Discrete Data)

এই অধ্যায় শুরু করার আগে আমরা কন্টিনিউয়াস ডেটা (Continuous Data) ও ডিসক্রিট ডেটা (Discrete Data) সম্পর্কে একটু জানব। সহজ করে বলতে গেলে, কন্টিনিউয়াস ডেটা হচ্ছে একটি নির্দিষ্ট সীমার মধ্যেকার যে-কোনো সংখ্যা। যেমন, আমরা যদি চিন্তা করি যে ০ ও ৩-এর ভেতরে যে-কোনো সংখ্যা এবং যদি বলে দিই আমরা কন্টিনিউয়াস মান নিয়ে কাজ করছি, তাহলে আমরা এই ০ ও ৩-এর ভেতরের যে-কোনো সংখ্যা নিয়ে কাজ করতে পারি, হোক সেটি পূর্ণ সংখ্যা (শুধু ০, ১, ২, ৩ ইত্যাদি), কিংবা দশমিকযুক্ত সংখ্যা (০.০০১, ০.০০০০৩৪৫, ১.২৩৪৪৩৪৪৫৬ ইত্যাদি)। বুঝতেই পারছেন, যদি আমরা কন্টিনিউয়াস ডেটা নিয়ে কাজ করি, তাহলে ০ ও ৩-এর ভেতরে অসীমসংখ্যক সংখ্যা পাওয়া সম্ভব।

আবার, আমরা যদি ডিসক্রিট ডেটা (Discrete Data) নিয়ে কাজ করতে যাই, সেটি হবে কন্টিনিউয়াস ডেটার ঠিক বিপরীত। কন্টিনিউয়াস ডেটার ক্ষেত্রে যেমন আমরা একটি সীমার মধ্যেকার যে-কোনো সংখ্যা নিতে পারতাম, ডিসক্রিট ডেটার ক্ষেত্রে আমরা তা পারব না। আমরা শুধু আমাদের সীমার ভেতরের নির্দিষ্ট কিছু সংখ্যা নিয়ে কাজ করতে পারব। যেমন, 0 ও 3-এর এই সীমায় শুধু 0, 1, 2, 3 এই চারটি পূর্ণসংখ্যা নিয়েই কাজ করতে পারব।

এখন আসি আমাদের মূল কাজে। আমরা লিনিয়ার রিগ্রেশনের ক্ষেত্রে সব সময় কন্টিনিউয়াস ডেটা নিয়ে কাজ করি। রিগ্রেশন কথাটির মানেই অনেকটা এরকম বোঝায় যে আউটপুট হবে কোনো একটি কন্টিনিউয়াস মান।

এখন আমরা যে রিগ্রেশন শিখব, তার নাম লজিস্টিক রিগ্রেশন। কিন্তু, নামের মধ্যে রিগ্রেশন শব্দটি থাকা সত্ত্বেও এটি কন্টিনিউয়াস মান নয়, ডিসক্রিট মান আউটপুট দেয়। এটি কন্টিনিউয়াস বা ডিসক্রিট দুই ধরনের মান নিয়েই কাজ করতে পারে, কিন্তু এর আউটপুট সব সময় হয় ডিসক্রিট মান। এটি মূলত একটি ক্লাসিফিকেশন অ্যালগরিদম।

পরিচেদ ৪.২ : লজিস্টিক রিগ্রেশন (Logistic Regression)-এর হাইপোথিসিস

লজিস্টিক রিগ্রেশনে আমাদের লক্ষ্য হবে, হাইপোথিসিস থেকে পাওয়া আউটপুট যেন 0 থেকে 1-এর ভেতরে থাকে। আউটপুট যদি 0.5-এর চেয়ে ছোটো হয়, তাহলে আমাদের লজিস্টিক রিগ্রেশনের সর্বশেষ আউটপুট হবে 0, আর যদি তা 0.5-এর সমান বা এর চেয়ে বড়ো হয়, তাহলে লজিস্টিক রিগ্রেশনের সর্বশেষ আউটপুট হবে 1।

এর মান 0 থেকে 1-এর ভেতরে রাখার পেছনে যুক্তি হচ্ছে, এটি মূলত সন্তাব্যতার মান আউটপুট হিসেবে দেয়। ধরুন, আপনি একটি প্রাণীর ছবি ইনপুট দিয়েছেন এবং লজিস্টিক রিগ্রেশনের মাধ্যমে বের করতে চাইছেন যে, সেটি একটি কুকুরের ছবি কি না। যদি সেটি কুকুরের ছবি হয়, তাহলে হিসাবনিকাশ করে হাইপোথিসিস আউটপুট দেবে 0.5 থেকে 1-এর মধ্যবর্তী কোনো একটি মান যেটি বোঝাচ্ছে যে ছবিটি কুকুরের হওয়ার সন্তাব্যতা বেশি। আর যেহেতু, এই হাইপোথিসিসের মান 0.5 থেকে 1-এর মধ্যে আছে, সেহেতু অ্যালগরিদমের ফাইনাল আউটপুট হবে 1, অর্থাৎ ছবিটি একটি কুকুরের ছবি।

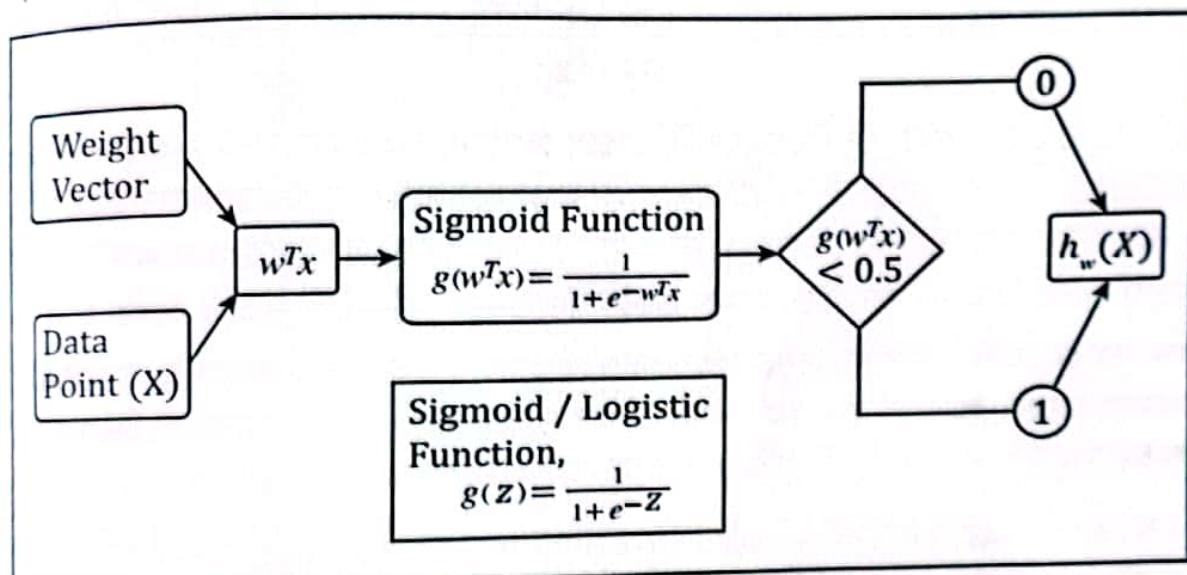
আর যদি সেটি না হয়, অর্থাৎ ছবিটি যদি কুকুরের না হয়ে অন্য কিছুর হয়, তাহলে আপনার হাইপোথিসিস হিসাবনিকাশ করে আউটপুট দেবে 0 থেকে 0.5-এর মধ্যবর্তী কোনো একটি সংখ্যা (0.5 হতে পারবে না, এর চেয়ে ছোটো হতে হবে কিন্তু) যেটি বোঝাচ্ছে যে ছবিটি কুকুরের হওয়ার

অধ্যায় ৪ : লজিস্টিক রিগ্রেশন (Logistic Regression)

সম্ভাব্যতা অনেক কম এবং সেটি থেকে পরবর্তী সময়ে অ্যালগরিদমের ফাইনাল আউটপুট হবে ০
অর্থাৎ ছবিটি কুকুরের নয়।

একটি খেয়াল রাখবেন, এখানে আমরা দুই ধরনের আউটপুটের কথা বলেছি, একটি হাইপোথিসিসের আউটপুট, আরেকটি লজিস্টিক রিগ্রেশনের আউটপুট। দুটোর মধ্যে পার্দক্ষ্য কী সেটি এখনই পরিষ্কার করে দিচ্ছি।

নিয়ার রিগ্রেশনের ক্ষেত্রে, আমাদের হাইপোথিসিসের সমীকরণে প্যারামিটার ও x -এর বিভিন্ন মান বসিয়ে হিসাব করে যে ফলাফল আসত, সেটিই ছিল আমাদের সর্বশেষ আউটপুট। লজিস্টিক রিগ্রেশনের ক্ষেত্রে তা হবে না। হাইপোথিসিসের আউটপুট ও অ্যালগরিদমের আউটপুট আলাদা হবে। নিচের ফ্রোচার্ট লক্ষ করুন (ফ্রোচার্ট 4.2.1):



ফ্রোচার্ট 4.2.1

লজিস্টিক রিগ্রেশনে আমরা আগের মতোই $w^T x$ -এর মান হিসাব করব। এর পরে, এই $w^T x$ -এর মান আমরা একটি সিগময়েড (Sigmoid) বা লজিস্টিক (Logistic) ফাংশনে ইনপুট হিসেবে দেব। সিগময়েড/লজিস্টিক ফাংশনটি হবে এরকম –

$$g(z) = \frac{1}{1 + e^{-z}}$$

এখানে $z = w^T x$ হবে। ছবি 4.2.2-তে একটি সিগময়েড ফাংশন দেখানো হলো। এটি থেকে আমরা যে আউটপুট পাব, সেটিই আমাদের হাইপোথিসিস $h_w(x)$ -এর মান। এখন আমরা দেখব, যে এই সিগময়েড ফাংশন যে মান দিচ্ছে, সেটি 0.5-এর চেয়ে ছোটো কি না (অতুকু নিশ্চিত ধারুন যে সিগময়েড ফাংশনের আউটপুটের মান 0 থেকে 1-এর ভেতরেই থাকবে)। যদি ছোটো

হয়, তাহলে আমাদের $h_w(x)$ হবে ০। আর যদি না হয়, অর্থাৎ যদি ০.৫-এর চেয়ে বড়ো বা এর সমান হয়, তাহলে আমাদের $h_w(x)$ হবে ১।

নিচে আপনাদের বোঝার সুবিধার্থে একটি টেবিল আকারে দেওয়া হলো :

$w^T x$	$g(w^T x) = \frac{1}{1 + e^{-w^T x}}$ (এটি হাইপোথিসিস থেকে প্রাপ্ত মান)	$h_w(x)$ (এটি লজিস্টিক রিগ্রেশনের ফাইনাল আউটপুট)
0	0.5	1
> 0	> 0.5	1
< 0	< 0.5	0

টেবিল 4.2.1

ছবি 4.2.2 থেকে একটু যদি বিশ্লেষণ করি, তাহলে দেখবেন, z -এর মান যতই $-\infty$ (negative infinity)-এর দিকে যাবে, e^{-z} -এর মান ততই বড়ো হতে থাকবে (ঝণাত্মকে মিলে ধনাত্মক হয়ে যাবে), অর্থাৎ ভগ্নাংশের হরের মান $1 + e^{-z}$ বাড়তে থাকবে এবং একটি বিশাল সংখ্যায় পরিণত হবে। তার মানে আমরা হাতে পাছি, $\frac{1}{\text{বিশাল কোনো একটি সংখ্যা}}$ এবং আমরা জানি ১-কে আমরা যত বিশাল সংখ্যা দিয়ে ভাগ করব, তার ভাগফল তত শূন্যের কাছাকাছি হবে। সেটিই আমরা গ্রাফে দেখতে পাচ্ছি। যতই আমরা z -এর মান কমাব (x -অক্ষ বরাবর বাঁ দিকে যাব), ততই আমরা শূন্যের কাছাকাছি যাব এবং একসময় শূন্যে পৌঁছব।

একইভাবে, z -এর মান যতই $+\infty$ (positive infinity)-এর দিকে যাবে, e^{-z} -এর মান ততই ছোটো হতে থাকবে এবং শূন্যের কাছাকাছি যেতে থাকবে, অর্থাৎ ভগ্নাংশের হরের মান $1 + e^{-z}$ তখন ১-এর কাছাকাছি যেতে থাকবে যেহেতু,

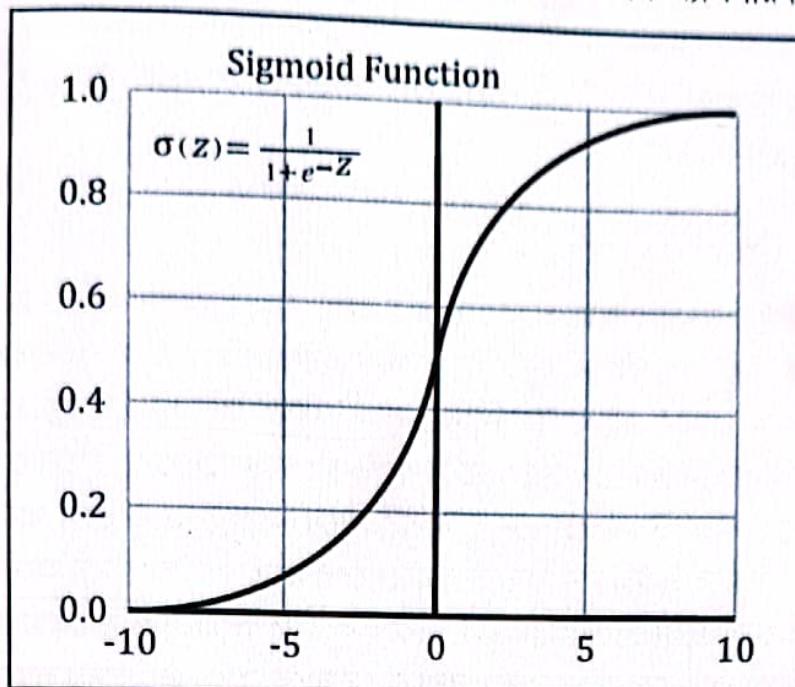
$$1 + \text{খুব ক্ষুদ্র কোনো সংখ্যা} (\text{শূন্যের কাছাকাছি}) \approx 1$$

তার মানে এবারে আমরা হাতে পাছি, $\frac{1}{1 - \text{এর খুব কাছাকাছি কোনো সংখ্যা}} (\text{যেমন } 1.00001)$ যার ফলে শেষমেশ আমাদের ভাগফল দাঁড়াচ্ছে ১-এর খুব কাছাকাছি কোনো মান অর্থাৎ, ১-ই ধরে নেওয়া যায়। সেটিই আমরা গ্রাফে দেখতে পাচ্ছি। যতই আমরা z -এর মান বাঢ়াব (x -অক্ষ বরাবর ডানদিকে যাব), ততই আমরা ১-এর কাছাকাছি যাব এবং একসময় ১-এ পৌঁছব।

এখন শেষ কথা হচ্ছে – এটাকে তাহলে লজিস্টিক ক্লাসিফিকেশন কেন বলা হলো না, কেন রিগ্রেশনই বলা হলো একে? এর উত্তর খুব সহজ। একটু খেয়াল করলে দেখবেন, আমরা আমাদের হাইপোথিসিস থেকে যে মান পাচ্ছি, সেটি কিন্তু একটি কন্টিনিউয়াস মান, তাই না? আর আমরা

অধ্যায় ৪ : লজিস্টিক রিগ্রেশন (Logistic Regression)

আউটপুট হিসেবে কন্টিনিউয়াস মান কোথায় পেতাম? রিগ্রেশনের ক্ষেত্রে, ঠিক? আর তা-ই একে লজিস্টিক রিগ্রেশন বলা হয়। কিন্তু এটি মূলত একটি ক্লাসিফিকেশন অ্যালগরিদম।



গ্রাফ 4.2.2

লজিস্টিক রিগ্রেশনের আরেকটি বৈশিষ্ট্য হচ্ছে এটি একটি টু-ক্লাস ক্লাসিফিকেশন অ্যালগরিদম (Two-class classification algorithm)। অর্থাৎ, এটি কোন ছবি – কুকুরের কিংবা কুকুরের নয় – এ দুটোর মধ্যে তফাত বের করতে পারবে। কিংবা, ছবিটি কুকুরের নাকি বিড়ালের – এরকম দুটি ক্লাসের মধ্যে ক্লাসিফিকেশন করতে পারবে।

এখন যদি বলা হতো কুকুর, বিড়াল ও হরিণ – এই তিনটি ক্লাসের মধ্যে ক্লাসিফিকেশন করতে, তখন কিন্তু এই সাধারণ লজিস্টিক রিগ্রেশন সেটি পারত না। তার জন্য দরকার হতো মাল্টিনোমিয়াল লজিস্টিক রিগ্রেশন (Multinomial Logistic Regression), যেটি এই সাধারণ লজিস্টিক রিগ্রেশনেরই আরেকটি ভার্শন। সেটি আপাতত আমাদের বইয়ের এখতিয়ারের বাইরে থাকছে।

লজিস্টিক রিগ্রেশনের আরেকটি বৈশিষ্ট্য এখানে বলে রাখি। লজিস্টিক রিগ্রেশন তখনই কাজ করে, যদি আমাদের ডেটা Higher Dimension-এ নিয়ে গেলে সেটি লিনিয়ারলি সেপারেবল (Linearly Separable) হয়। এর বিস্তারিত নিয়ে আর এখানে বলব না।

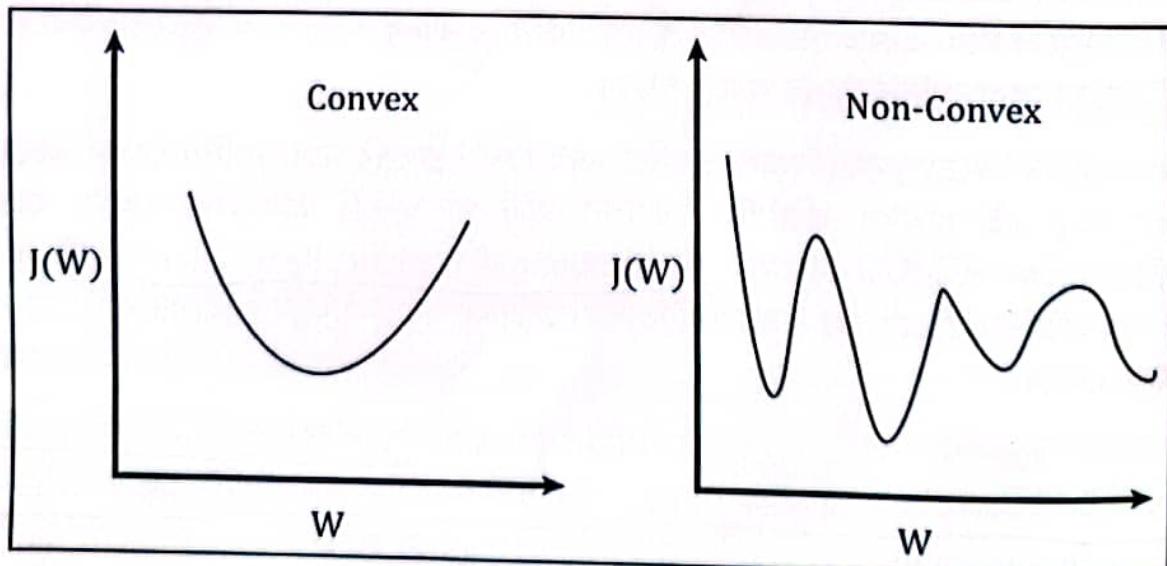
প্রবর্তী অধ্যায়েই (সাপোর্ট ভেক্টর মেশিন – Support Vector Machine) পরিচ্ছেদ ৫.৩-এ এই সম্পর্কে বিস্তারিত বলা হয়েছে।

পরিচেদ 4.3 : লজিস্টিক রিপ্রেশনের কস্ট ফাংশন (Cost Function) ও প্রয়োগ পদ্ধতি

এখন আমরা আমাদের লজিস্টিক রিপ্রেশনের কস্ট ফাংশন নিয়ে কথা বলব। লিনিয়ার রিপ্রেশনে আমাদের কস্ট ফাংশন ছিল –

$$\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

কিন্তু, লজিস্টিক রিপ্রেশনের ক্ষেত্রে আমাদের এই কস্ট ফাংশনে পরিবর্তন আসবে। এর কারণ হচ্ছে, আমরা লিনিয়ার রিপ্রেশনের কস্ট ফাংশনে হাইপোথিসিস $h_w(x)$ ব্যবহার করছি এবং লজিস্টিক রিপ্রেশনের ক্ষেত্রে এই হাইপোথিসিস একটি সিগময়েড ফাংশন হয়, তাই না (ফ্লো-চার্ট দ্রষ্টব্য)? এখন, আমরা যদি লিনিয়ার রিপ্রেশনের কস্ট ফাংশন লজিস্টিক রিপ্রেশনেও ব্যবহার করি, এর ফলে যেটি হবে, লজিস্টিক রিপ্রেশনে হাইপোথিসিসে সিগময়েড ফাংশন থাকার কারণে কস্ট ফাংশনের চেহারা দাঁড়াবে একটি নন-কনভেক্স ফাংশনের মতো (যে ফাংশনে অনেকগুলো লোকাল মিনিমা আছে)। অনেকগুলো লোকাল মিনিমা থাকার কারণে, আমাদের যে-কোনো একটি লোকাল মিনিমাম পর্যন্তে আটকা পড়ে যাওয়ার সম্ভাবনা বেড়ে যাবে, অথচ আমাদের লক্ষ্য হলো সব সময় কস্ট ফাংশনের সর্বনিম্ন মান অর্থাৎ গ্লোবাল মিনিমামে পৌঁছানোর। সেজন্য আমরা লজিস্টিক রিপ্রেশনে লিনিয়ার রিপ্রেশনের কস্ট ফাংশন ব্যবহার না করে নতুন একটি কস্ট ফাংশন ব্যবহার করব। ব্যাপারটি বোঝার জন্য ছবি 4.3.1-এর গ্রাফ দৃঢ়ি দেখুন।



ছবি 4.3.1

সূতরাং, লজিস্টিক রিপ্রেশনের জন্য নতুন কস্ট ফাংশন ব্যবহার করব,

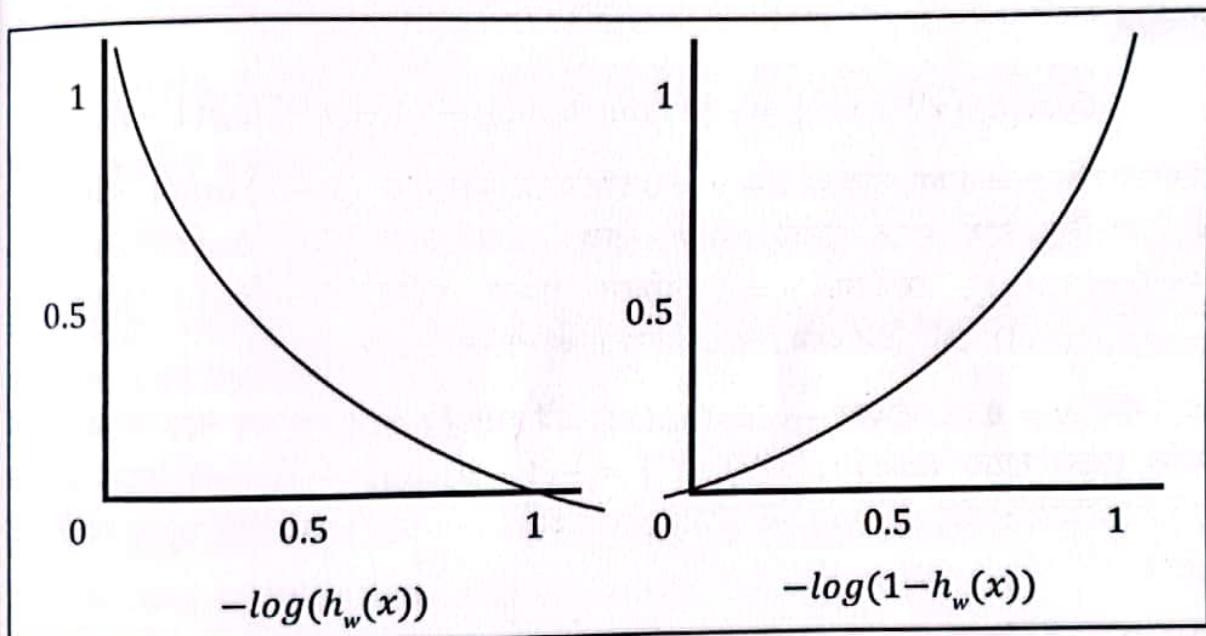
অধ্যায় 8 : লজিস্টিক রিফ্রেশন (Logistic Regression)

$$J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \text{Cost}\{h_w(x^{(i)}), y^{(i)}\}$$

যেখানে,

$$\text{Cost}\{h_w(x^{(i)}), y^{(i)}\} = \begin{cases} -\log(h_w(x)), & \text{if } y = 1 \\ -\log(1 - h_w(x)), & \text{if } y = 0 \end{cases}$$

প্রথমেই বলে রাখি, এই y হচ্ছে আগের মতোই, আমাদের আউটপুটের প্রকৃত মান যেটি আমরা আমাদের ট্রেনিং ডেটা থেকে পাব। এখানে দেখুন, যদি $y = 1$ হয়, তাহলে $\text{Cost}\{h_w(x^{(i)}), y^{(i)}\} = -\log(h_w(x))$ হবে, আবার যদি $y = 0$ হয় তাহলে, $\text{Cost}\{h_w(x^{(i)}), y^{(i)}\} = -\log(1 - h_w(x))$ হবে। ছবি 4.3.2-এর গ্রাফ থেকে ফাংশনগুলো দেখতে কী রকম তার একটি ধারণা পাওয়া যাবে।



ছবি 4.3.2

Y	$h_w(x)$	Cost
0	0	0
	$\cong 1$	$\cong \infty$
1	1	0
	$\cong 0$	$\cong \infty$

টেবিল 4.3.1

আপনারা ছবি 4.3.2-এর কস্ট ফাংশনের গ্রাফ দুটি পর্যবেক্ষণ করলে, টেবিল 4.3.1-এর অনুবন্ধে পুরুত্বে পারবেন। গ্রাফ ও চার্ট অনুযায়ী, আমাদের প্রকৃত আউটপুট যদি 1 হয় এবং আমরা হাইপোথিসিস থেকে প্রেডিকশন করেও যদি 1 পাই, তাহলে আমাদের কস্ট হবে শূন্য; যেহেতু প্রকৃত মান এবং আমাদের অনুমান করা মান মিলে গেছে। যতই হাইপোথিসিসের মান শূন্যের কাছাকাছি যেতে থাকবে, ততই কস্ট-এর মান অসীমের দিকে ধাবিত হবে। একই পদ্ধতি $y = 0$ এর ক্ষেত্রেও প্রযোজ্য।

আমরা চাইলে আমাদের কস্ট ফাংশনটিকে আরেকটু সহজ করে, দুটি শর্তই একটি সমীকরণের মধ্যে নিয়ে এসে প্রকাশ করতে পারি। সে ক্ষেত্রে, কস্ট ফাংশনটি হবে এরকম :

$$J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m Cost\{h_w(x^{(i)}), y^{(i)}\}$$

যেখানে,

$$Cost\{h_w(x^{(i)}), y^{(i)}\} = -y^{(i)} \log(h_w(x)) - (1 - y^{(i)}) \log(1 - h_w(x))$$

এখানে, যদি $y = 1$ হয়, তাহলে, $1 - y = 0$ হয়ে যাচ্ছে, যার মানে $(1 - y) \log(1 - h_w(x))$ এ অংশটুকু বাদ পড়ে যাচ্ছে। ফলে বাকি থেকে যাচ্ছে $Cost\{h_w(x^{(i)}), y^{(i)}\} = -y \log(h_w(x))$, যেখানে $y = 1$ বসিয়ে আমরা পাচ্ছি $Cost\{h_w(x^{(i)}), y^{(i)}\} = -\log(h_w(x))$, যেটি ছিল আমাদের প্রথম কস্ট ফাংশনের প্রথম প্রেক্ষাপট।

আর যদি, $y = 0$ হয়, তাহলে $-y \log(h_w(x))$ এই অংশটুকু পুরোটুকু বাদ পড়ে যাচ্ছে। ফলে বাকি থেকে যাচ্ছে $Cost\{h_w(x^{(i)}), y^{(i)}\} = -(1 - y) \log(1 - h_w(x))$ যাতে $y = 0$ বসিয়ে আমরা পাচ্ছি $-(1 - y) \log(1 - h_w(x))$, যেটি হলো আমাদের প্রথম কস্ট ফাংশনের দ্বিতীয় প্রেক্ষাপট।

তাহলে আমরা দেখতে পাচ্ছি, দ্বিতীয় কস্ট ফাংশনটিতে আসলে একটি সমীকরণ দিয়েই প্রথম কস্ট ফাংশনের দুটি প্রেক্ষাপটই আমরা পেয়ে যাচ্ছি। তাই, আমরা কস্ট ফাংশন হিসেবে দ্বিতীয়টিই ব্যবহার করব।

যাহোক, আমরা আমাদের কস্ট ফাংশন পেয়ে গেলাম। আর বাকি থাকল এখন শুধু প্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম কিংবা নরমাল ইকুয়েশন (Normal Equation) পদ্ধতিতে প্যারামিটার আপডেট করা যেটি ছবছ আগের মতোই হবে, যেভাবে আমরা মাল্টিভ্যারিয়েট লিনিয়ার রিগ্রেশনের ক্ষেত্রে পড়ে এসেছি (পরিচ্ছেদ 3.8)।

প্রেডিয়েন্ট ডিসেন্ট আসলেই কীভাবে কাজ করে সেটি উদাহরণের সাহায্যে বুঝতে পরিচ্ছেদ 3.8 পড়ে নিতে পারেন। সেখানে লিনিয়ার রিগ্রেশনের ক্ষেত্রে কীভাবে প্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম

অধ্যায় ৪ : লজিস্টিক রিগ্রেশন (Logistic Regression)

কাজ করে, সেটি একেবারে হাতে কলমে দেখানো হয়েছে। এখানেও সেটি একইভাবে কাজ করবে।

পরিচেদ ৪.৮ : লজিস্টিক রিগ্রেশনে রেগুলারাইজেশনের ব্যবহার

লজিস্টিক রিগ্রেশনের প্রাথমিক বিষয়গুলো আমরা ইতিমধ্যেই পড়ে ফেলেছি। এটি খুবই জনপ্রিয় একটি ক্লাসিফিকেশন অ্যালগরিদম। লজিস্টিক রিগ্রেশনের কস্ট ফাংশন এরকম :

$$J(W) = \frac{1}{m} \sum_{i=1}^m \text{Cost}\{h_w(x^{(i)}), y^{(i)}\}$$

যেখানে,

$$\text{Cost}\{h_w(x^{(i)}), y^{(i)}\} = -y \log(h_w(x)) - (1-y) \log(1-h_w(x))$$

সুতরাং, পুরোটুকু একসঙ্গে মিলিয়ে লিখলে দাঁড়ায়,

$$J(W) = -\frac{1}{m} \left[\sum_{i=1}^m \{ y^{(i)} \log(h_w(x^{(i)})) + (1-y^{(i)}) \log(1-h_w(x^{(i)})) \} \right]$$

আমরা ইতিমধ্যেই পরিচেদ ৩.৭ থেকে জানি, L2 রেগুলারাইজেশনের জন্য আমাদেরকে এখানে অতিরিক্ত একটি টার্ম যোগ করতে হবে, সেটি হচ্ছে –

$$\frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

এখানে λ হচ্ছে আমাদের রেগুলারাইজেশন প্যারামিটার এবং আমরা রেগুলারাইজ করব w -কে অর্থাৎ আমাদের ওয়েইটগুলোকে। আমরা আরো জানি, আমরা যে পদ্ধতিতে রেগুলারাইজেশন করলাম, তার নাম হচ্ছে L2 রেগুলারাইজেশন। এটি ছাড়াও, আরেকটি পদ্ধতি আছে, যাকে বলে L1 রেগুলারাইজেশন। সেখানে আমাদেরকে যোগ করতে হতো –

$$\frac{\lambda}{2m} \sum_{j=1}^n |w_j|$$

আমরা আমাদের এই বইতে L2 রেগুলারাইজেশন ব্যবহার করব, কেননা ওভারফিটিং এডানোর জন্য L1 রেগুলারাইজেশনের চেয়ে L2 রেগুলারাইজেশন ভালো কাজ করে। এর আরো বিস্তারিত গাণিতিক হিসাবকিতাব আছে, যেগুলো সংগত কারণেই আমরা আপাতত এড়িয়ে যাব, পরবর্তী সময়ে যদি কাজ করতে করতে জানার প্রয়োজন হয় তখন জেনে নেব।

তাহলে L2 রেগুলারাইজেশন ব্যবহার করে আমাদের সর্বশেষ কস্ট ফাংশনটি দাঁড়াচ্ছে :

মেশিন লার্নিং অ্যালগরিদম

$$J(W) = -\frac{1}{m} \left[\sum_{i=1}^m \left\{ \begin{array}{l} y^{(i)} \log(h_w(x^{(i)})) \\ + (1 - y^{(i)}) \log(1 - h_w(x^{(i)})) \end{array} \right\} \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

দেখে হয়তো মনে হচ্ছে অনেক জটিল, কীভাবে কী করব? কীভাবে এটি যোগ করার পরে আমাদের গ্রেডিয়েন্ট ডিসেন্ট পদ্ধতি ব্যবহার করে প্যারামিটার আপডেট হবে?

আমরা গ্রেডিয়েন্ট ডিসেন্ট ব্যবহার করে কীভাবে লিনিয়ার রিফ্রেশনের প্যারামিটার বা ওয়েইটগুলো খুঁজে বের করেছিলাম, আশা করি সবার মনে আছে। তবু এখানে আমি আরেকবার লিখে দিচ্ছি:

প্রথম প্যারামিটারের জন্য :

$$\begin{aligned} w_0 &= w_0 - \alpha \frac{\partial}{\partial w_0} J(W) \\ \frac{\partial}{\partial w_0} J(W) &= \frac{\partial}{\partial w_0} \left[\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 \right] \\ &= \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_0^{(i)} \end{aligned}$$

পরবর্তী সব প্যারামিটার ($j = 1, 2, 3, \dots, n$) জন্য –

$$\begin{aligned} w_j &= w_j - \alpha \frac{\partial}{\partial w_j} J(W) \\ \frac{\partial}{\partial w_j} J(W) &= \frac{\partial}{\partial w_j} \left[\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 \right] \\ &= \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_j^{(i)} \end{aligned}$$

এখন, এটি ছিল সাধারণ গ্রেডিয়েন্ট ডিসেন্ট, কোনোরকম রেগুলারাইজেশন ছাড়াই। এখন, আমরা যদি রেগুলারাইজড কন্ট ফাংশন ব্যবহার করি, যেটি কিনা এরকম :

$$J(W) = -\frac{1}{m} \left[\sum_{i=1}^m \left\{ \begin{array}{l} y^{(i)} \log(h_w(x^{(i)})) \\ + (1 - y^{(i)}) \log(1 - h_w(x^{(i)})) \end{array} \right\} \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

সে ক্ষেত্রে, আমাদের প্যারামিটার আপডেটের ক্ষেত্রে সামান্য কিছু পরিবর্তন আসবে। প্রথম প্যারামিটার w_0 -এর জন্য আগে যা ছিল তা-ই থাকবে –

$$\begin{aligned} w_0 &= w_0 - \alpha \frac{\partial}{\partial w_0} J(W) \\ \frac{\partial}{\partial w_0} J(W) &= \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_0^{(i)} \end{aligned}$$

অধ্যায় ৪ : লজিস্টিক রিফ্রেশন (Logistic Regression)

প্রত্যেক প্যারামিটার ($j = 1, 2, 3, \dots, n$)-গুলোর জন্য পরিবর্তিত হয়ে হবে -

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(W)$$

$$\frac{\partial}{\partial w_j} J(W) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_j^{(i)} - \frac{\lambda}{m} w_j$$

বস, শুধু এটুকুই পরিবর্তন। বাকি সবকিছু আমরা যে লজিস্টিক রিফ্রেশন পড়েছি, সে পদ্ধতিতেই করতে হবে।

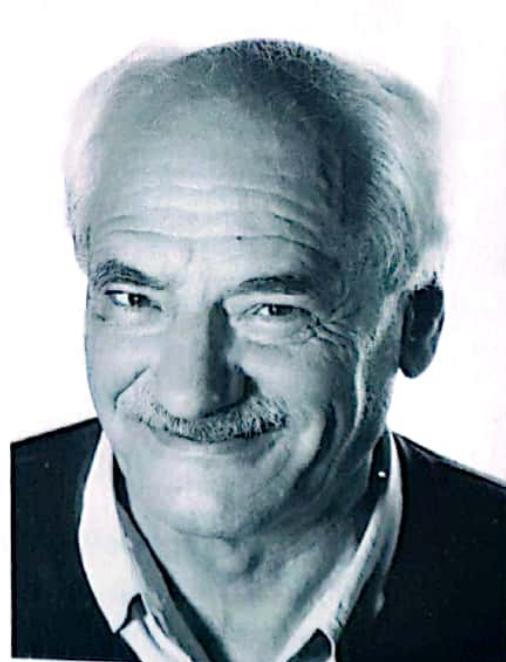
অধ্যায় ৫ : সাপোর্ট ভেস্ট্র মেশিন (Support Vector Machine - SVM)

সাপোর্ট ভেস্ট্র মেশিন অ্যালগরিদমটি প্রথম আবিষ্কার করেন ভ্লাদিমির (Vladimir Vapnik, 1936) ও আলেক্সেই (Alexey Chervonenkis, 1938 - 2014) নামের দুজন বিজ্ঞানী, 1963 সালে।

পরবর্তী সময়ে বোসের (Bernhard E. Boser), ইসাবেল (Isabelle M. Guyon) ও ভ্লাদিমির মিলে 1992 সালে আধুনিক সাপোর্ট ভেস্ট্র মেশিন ও কার্নাল ট্রিক সম্পর্কে ধারণা দেন, যা নন-লিনিয়ার ডেটা ক্লাসিফাই করার জন্য পরবর্তীতে ব্যবহৃত হয়।



Vladimir Vapnik, 1936



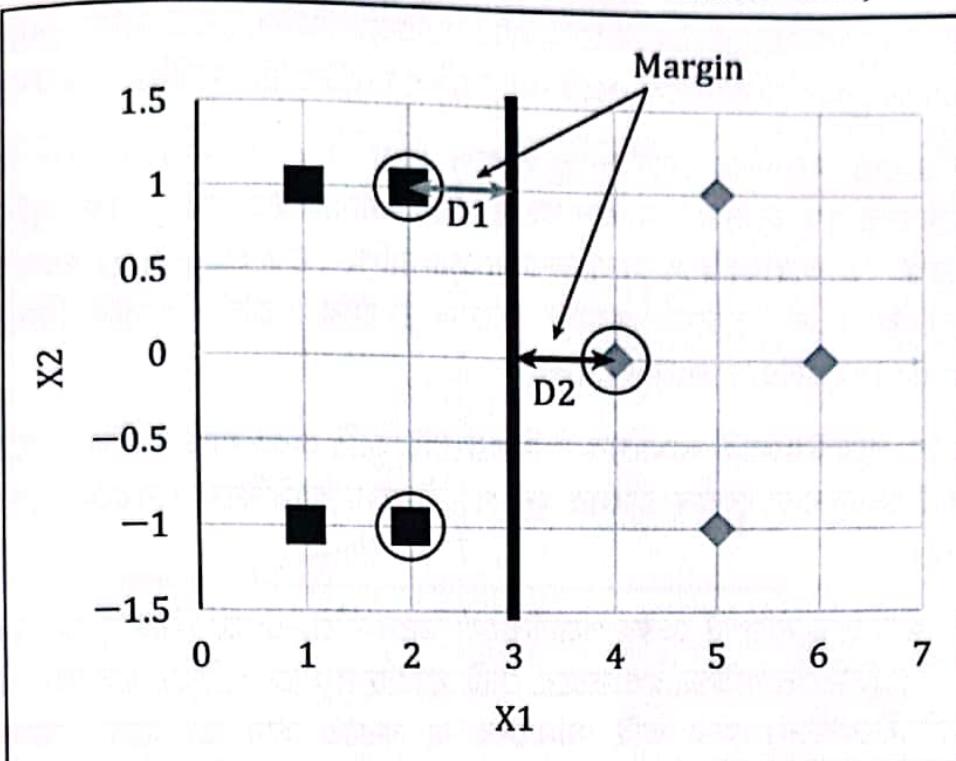
Alexey Chervonenkis, 1938 - 2014

ছবি 5.1

সাপোর্ট ভেস্ট্র মেশিন খুব সন্তুষ্ট মেশিন লার্নিংয়ের সবচেয়ে বহুল ব্যবহৃত এবং অন্যতম জনপ্রিয় অ্যালগরিদম। এই অ্যালগরিদম বহু জায়গায়, বহু প্রবলেমসেটে বহুভাবে ব্যবহৃত হয়েছে। এর জন্য অসংখ্য অপটিমাইজেশন মেথড রয়েছে, সেই সঙ্গে রয়েছে এর নানাবিধ অ্যাপ্লিকেশন।

সাপোর্ট ভেস্ট্র মেশিনের উদাহরণসহ ব্যবহারের পদ্ধতি শেখার আগে তাই আমরা এর গঠন সম্পর্কে প্রথমে একটুখানি জেনে নেব।

অধ্যায় ৫ : সাপোর্ট ভেক্টর মেশিন (Support Vector Machine - SVM)



ছবি 5.2

ওপরের গ্রাফটি (ছবি 5.2) লক্ষ করি। ধরি, বর্গাকৃতির চিহ্ন দিয়ে দেখানো চারটি পয়েন্ট Class 1-এর এবং ডায়মন্ড আকৃতির চিহ্ন দিয়ে দেখানো চারটি পয়েন্ট Class 2-এর।

এখন, যদি আমরা Class 1-এর প্রতিটি পয়েন্টের সঙ্গে Class 2-এর প্রতিটি পয়েন্টের দূরত্ব বের করি এবং সেখান থেকে বের করার চেষ্টা করি যে, এই দুই ক্লাসের কোন কোন পয়েন্টগুলো একে অপরের সবচেয়ে কাছাকাছি আছে (যে দুটো পয়েন্টের মধ্যে দূরত্ব মাপব, তারা অবশ্যই ভিন্ন ভিন্ন ক্লাসের হতে হবে), তখন দেখব যে $(2, 1), (2, -1)$ ও $(4, 0)$ – এই তিনটি পয়েন্ট একে অপরের সবচেয়ে কাছাকাছি (গ্রাফে বৃত্তাকৃতির চিহ্ন দিয়ে দেখানো হয়েছে)।

আরো লক্ষ করি, আমি যদি কোনো একটি রেখা টেনে এই দুটি ক্লাসকে আলাদা করতে চাই (মাঝখানে মোটা দাগ দিয়ে দেখানো, একে Decision Boundary বলে), তবে দেখুন যে $(2, 1), (2, -1)$ ও $(4, 0)$ এই তিনটি পয়েন্টই ডিসিশন বাউন্ডারির সবচেয়ে কাছে থাকবে।

আর তাই এই পয়েন্টগুলোকে বলা হয় সাপোর্ট ভেক্টর। গ্রাফটি ভালো করলে দেখা যাবে যে, $(2, 1)$ ও $(2, -1)$ হচ্ছে ডিসিশন বাউন্ডারি থেকে Class 1-এর সবচেয়ে কাছের দুটি পয়েন্ট এবং এদের থেকে ডিসিশন বাউন্ডারির দূরত্ব D_1 । তাই বলা যায়, Class 1-এর পয়েন্টগুলো থেকে এই ডিসিশন বাউন্ডারির সর্বনিম্ন দূরত্ব D_1 । একইভাবে, $(4, 0)$ হচ্ছে ডিসিশন বাউন্ডারি

থেকে Class 2-এর সবচেয়ে কাছের পয়েন্ট এবং এর থেকে ডিসিশন বাউন্ডারির দূরত্ব D₂। তাই বলা যায়, Class 2-এর পয়েন্টগুলো থেকে এই ডিসিশন বাউন্ডারির সর্বনিম্ন দূরত্ব D₂।

এখন, দুটি ক্লাসের মধ্যেকার মোট দূরত্ব তাহলে হলো $D = D_1 + D_2$ । একে আমরা বলি মার্জিন। আমাদের লক্ষ হচ্ছে এই মার্জিন যতটা স্ন্তুব ম্যাস্কিমাইজ করা। কারণ, দুটি ভিন্ন ভিন্ন ক্লাসের পয়েন্ট যত কাছাকাছি থাকবে, তত ওভারল্যাপিং (Overlapping) হওয়ার সন্তান বাড়বে এবং ফলে এক ক্লাসের পয়েন্ট আরেক ক্লাসের পয়েন্ট হিসেবে মিসক্লাসিফাইড (Misclassified) হওয়ার সন্তান বাড়বে।

সাপোর্ট ভেস্ট্র পয়েন্টগুলোই আমাদের ডিসিশন বাউন্ডারি এবং মার্জিন ঠিক করতে সহায়তা করে। সাপোর্ট ভেস্ট্র বাদে কোনো ক্লাসের অন্য পয়েন্টগুলো ক্লাসিফিকেশনে তেমন কোনো গুরুত্ব বহন করে না।

আমরা যদি আমাদের সাপোর্ট ভেস্ট্র পয়েন্টগুলো ডানে-বাঁয়ে-ওপরে-নিচে সরাই, তাহলে তাৰ সঙ্গে সঙ্গে কিন্তু ডিসিশন বাউন্ডারিও সরবে, যেটি মার্জিনের মানে পরিবর্তন আনবে এবং প্রভাব ফেলবে ক্লাসিফিকেশনে। আর তাই আমাদের যা করতে হবে তা হলো, সাপোর্ট ভেস্ট্র পয়েন্টগুলোর মধ্যবর্তী এমন কোনো জায়গায় ডিসিশন বাউন্ডারি আঁকতে হবে, যাতে মার্জিন বৃদ্ধি পায়। এই ডিসিশন বাউন্ডারি খুঁজে বের করার কাজই করে দেয় সাপোর্ট ভেস্ট্র মেশিন অ্যালগরিদম। সাপোর্ট ভেস্ট্র মেশিন সব সময় বৃহত্তম মার্জিন (Largest Margin) খুঁজে বের করার চেষ্টা করে দেখে একে লার্জেস্ট/ম্যাক্সিমাম মার্জিন ক্লাসিফায়ার (Largest/Maximum Margin Classifier)-ও বলা হয়।

পরিচ্ছেদ ৫.১ : লিনিয়ার সাপোর্ট ভেস্ট্র মেশিন (Linear Support Vector Machine - LSVM)

আমরা লজিস্টিক রিগ্রেশন পড়ার সময় হাইপোথিসিস হিসেবে নিয়েছিলাম –

$$h_w(x) = g(z) = \frac{1}{1+e^{-z}}$$

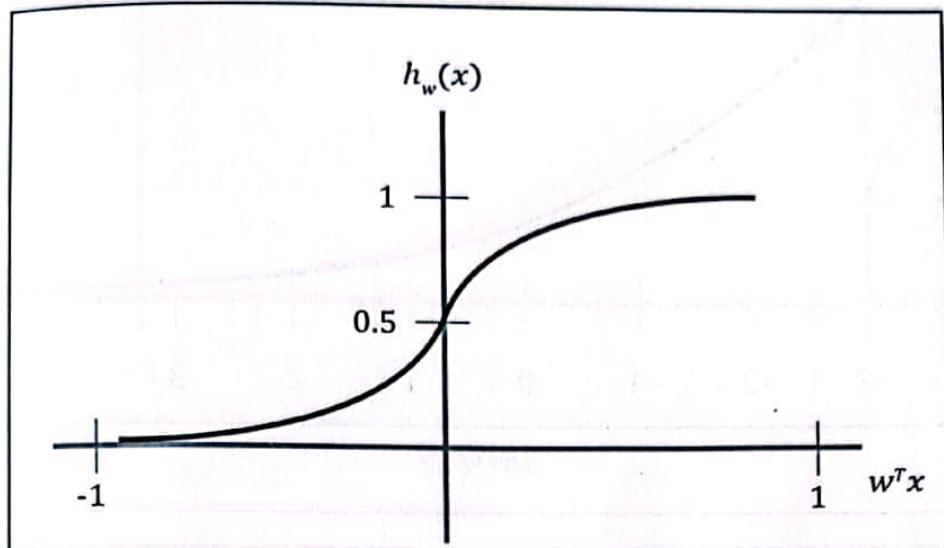
যেখানে, $z = W^T x$ ছিল। একে যদি আমরা একটি গ্রাফে প্লট করি, তাহলে গ্রাফ 5.1.1-এর মত একটি গ্রাফ পাব।

লজিস্টিক রিগ্রেশনের ক্ষেত্রে,

- যদি প্রকৃত আউটপুট $y = 1$ হয়, তাহলে আমরা চাই $h_w(x) \approx 1$ হোক, অর্থাৎ $W^T x \gg 0$ হোক। আবার,

অধ্যায় ৫ : সাপোর্ট ভেস্ট্র মেশিন (Support Vector Machine - SVM)

- যদি প্রকৃত আউটপুট $y = 0$ হয়, তাহলে আমরা চাই $h_w(x) \approx 0$ হোক, অর্থাৎ $W^T x \ll 0$ হোক।



গ্রাফ 5.1.1

এখন, আমরা যা করব তা হলো, লজিস্টিক রিগ্রেশনে যা আমরা পড়েছি, তাকেই কিছুটা পরিবর্তিত করে সেখান থেকে কীভাবে সাপোর্ট ভেস্ট্র মেশিন শেখা যায় তা দেখব।

লজিস্টিক রিগ্রেশনের ক্ষেত্রে, একটিমাত্র ডেটা পয়েন্টের জন্য Cost ছিল –

$$Cost = -y \log(h_w(x)) - (1 - y) \log(1 - h_w(x))$$

এখন, এই Cost ব্যবহার করে আমরা ওপরে যে দুটি প্রেক্ষাপট উল্লেখ করলাম ($y = 1$ ও $y = 0$) সেই দুটি নিয়ে একটু বিস্তারিত বিশ্লেষণ করি –

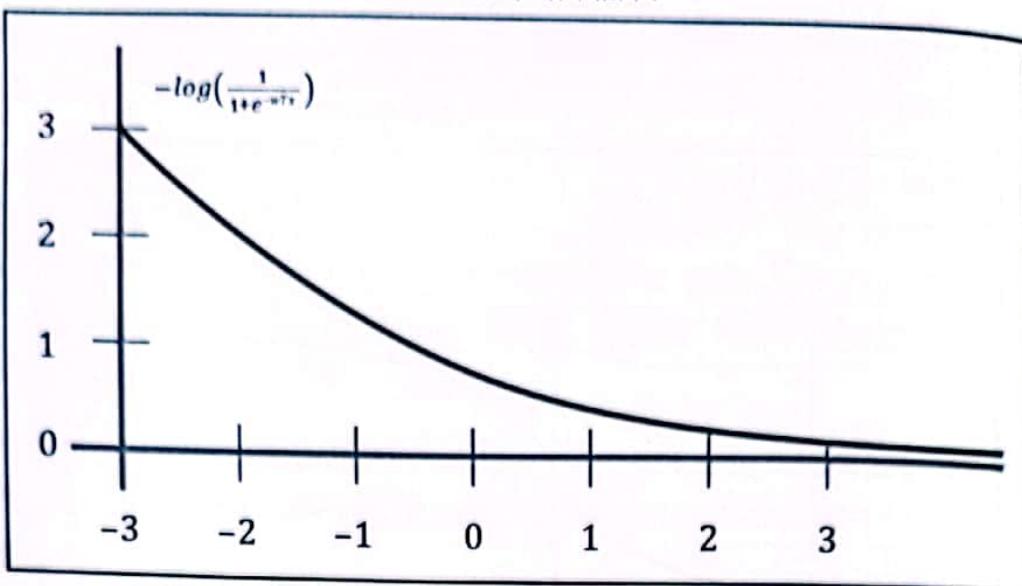
আমরা যদি, $y = 1$ এর জন্য, ($W^T x \gg 0$) নিই, সে ক্ষেত্রে আমাদের কস্ট দাঁড়ায়,

$$Cost = -\log(h_w(x)) = -\log\left(\frac{1}{1+e^{-W^T x}}\right)$$

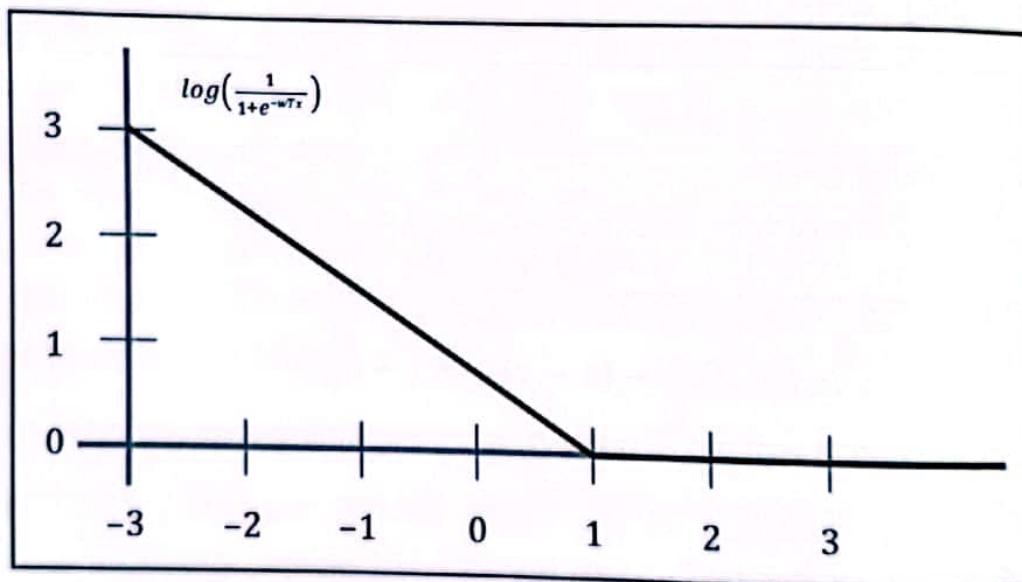
এখন, শুধু এই অংশটুকু কে যদি আমি গ্রাফে প্লট করি, তাহলে গ্রাফ 5.1.2-এর মত একটি গ্রাফ পাব।

এখন সাপোর্ট ভেস্ট্র মেশিন তৈরির জন্য, আমরা গ্রাফ 5.1.2-এর কার্ডটিকে দুই ভাগে ভাগ করব গ্রাফ 5.1.3-এর মতো।

এই গ্রাফ 5.1.3-এর সঙ্গে গ্রাফ 5.1.2-এর কার্ডের অনেকখানিই মিল আছে। শুধু পার্থক্য হলো – আগের গ্রাফের চেয়ে এই গ্রাফ একটু বেশি তীক্ষ্ণ।



গ্রাফ 5.1.2



গ্রাফ 5.1.3

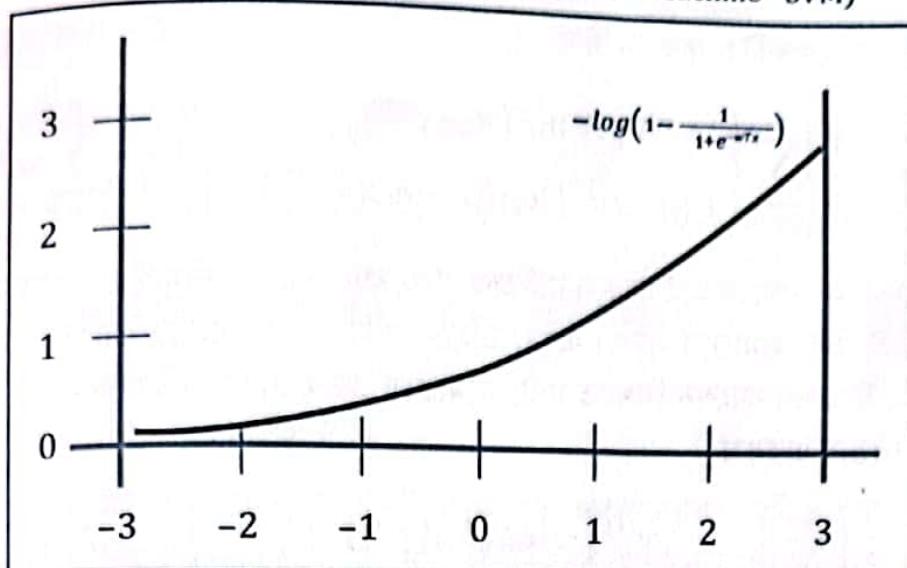
ঠিক একইভাবে, আমরা যদি $y = 0$ এর জন্য ($W^T x \ll 0$) নিই, সে ক্ষেত্রে আমাদের কস্ট দাঁড়ায়,

$$Cost = -\log(1 - h_w(x)) = -\log\left(1 - \frac{1}{1+e^{-W^T x}}\right)$$

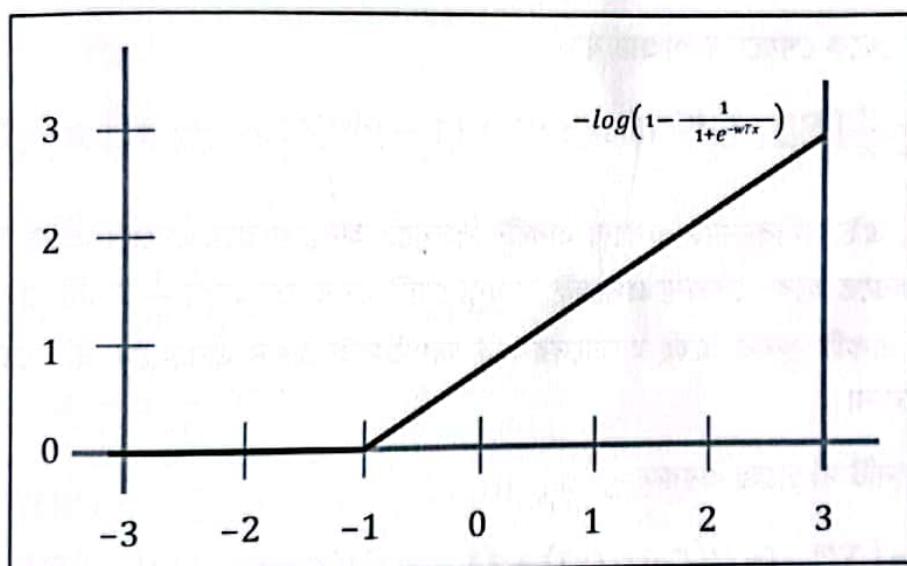
এখন, শুধু এই অংশটুকু যদি আমি গ্রাফে প্লট করি, তাহলে তাহলে গ্রাফ 5.1.4-এর মত একটি গ্রাফ পাব।

এখন আবার আগের মতোই আমরা গ্রাফ 5.1.4-এর কার্ডটিকে দুই ভাগে ভাগ করব গ্রাফ 5.1.5-এর মতো।

অধ্যায় ৫ : সাপোর্ট ভেক্টর মেশিন (Support Vector Machine - SVM)



গ্রাফ 5.1.4



গ্রাফ 5.1.5

এখন, ($y = 1$)-এর জন্য আমরা যে কস্ট পেলাম, ধরি তা হলো $Cost_1(z)$ ও ($y = 0$)-এর জন্য যে কস্ট পেলাম, ধরি তা হলো $Cost_0(z)$ ।

পরিচেদ 8.8 থেকে হয়তো সবার মনে আছে যে, আমরা লজিস্টিক রিগ্রেশনের জন্য রেগুলারাইজেশন ব্যবহার করে যে কস্ট ফাংশন পেয়েছিলাম তা ছিল এরকম :

$$J(W) = -\frac{1}{m} \left[\sum_{i=1}^m \left\{ y^{(i)} \log(h_w(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_w(x^{(i)})) \right\} \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

এবং আমাদের অবজেকটিভ ফাংশন ছিল –

$$\min_w -\frac{1}{m} \left[\sum_{i=1}^m \left\{ y^{(i)} \log(h_w(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_w(x^{(i)})) \right\} \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

অর্থাৎ, আমাদেরকে ওপরের এই রেজুলারাইজড কস্ট ফাংশনকে মিনিমাইজ করতে হবে W -এর সাপেক্ষে, এটিই ছিল আমাদের লক্ষ্য। এখন তাহলে, যদি সাপোর্ট ভেস্ট্র মেশিনের জন্য এরকম একটি অপটিমাইজেশন ফাংশন লিখতে যাই, আমাদের এতক্ষণের যাবতীয় কাজকর্মের সাপেক্ষে, তাহলে সেটি দাঁড়ায় এরকম –

$$\min_w \frac{1}{m} \left[\sum_{i=1}^m \left\{ y^{(i)} (-\log(h_w(x^{(i)}))) + (1 - y^{(i)}) (-\log(1 - h_w(x^{(i)}))) \right\} \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

এবং সেখান থেকে শেষমেশ পাওয়া যায় –

$$\min_w \frac{1}{m} [\sum_{i=1}^m \{y^{(i)}(Cost_1(z)) + (1 - y^{(i)})(Cost_0(z))\}] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

এখন দেখুন, এই সমীকরণটি এখনো একটি বিদ্যুটে অবস্থায় রয়ে গেছে। একে আরো একটু সরলীকরণ করতে হবে। সেজন্য প্রথমেই আমরা যেটি করব, তা হলো $\frac{1}{m}$ টার্মটি বাদ দিয়ে দেব, কেননা এটি একটি ধ্রুবক এবং আমাদের এই অপটিমাইজেশন প্রবলেমে এটি তেমন কোনে প্রভাব ফেলবে না।

তাহলে ফাংশনটি দাঁড়াচ্ছে এরকম –

$$\min_w [\sum_{i=1}^m \{y^{(i)}(Cost_1(z)) + (1 - y^{(i)})(Cost_0(z))\}] + \frac{\lambda}{2} \sum_{j=1}^n w_j^2$$

এখন এই ফাংশনটি আমরা এইভাবে লিখতে পারি –

$$A + \lambda B$$

যেখানে, $A = \min_w [\sum_{i=1}^m \{y^{(i)}(Cost_1(z)) + (1 - y^{(i)})(Cost_0(z))\}]$ এবং

$$B = \frac{1}{2} \sum_{j=1}^n w_j^2$$

এখন, ওপরের এই $A + \lambda B$ -কে আমরা একটু বদলে এইভাবেও লিখতে পারি,

অধ্যায় ৫ : সাপোর্ট ভেক্টর মেশিন (Support Vector Machine - SVM)

$$CA + B \text{ যেখানে, } C = \frac{1}{\lambda}$$

সুতরাং, সবশেষে আমাদের ফাইনাল অপটিমাইজেশন প্রবলেমটি দাঁড়ায় এরকম –

$$\min_w C \left[\sum_{i=1}^m \{y^{(i)}(Cost_1(z)) + (1 - y^{(i)})(Cost_0(z))\} \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

এবং আমাদের সাপোর্ট ভেক্টর মেশিনের জন্য হাইপোথিসিস হবে –

$$h_w(x) = \begin{cases} 0, & W^T x < 0 \\ 1, & W^T x \geq 0 \end{cases}$$

এখন আসি প্যারামিটার কীভাবে আপডেট করব সে আলোচনায়। এটিও আগের পরিচ্ছেদে দেখানো লজিস্টিক রিগ্রেশনের রেণুলারাইজড প্যারামিটার যেভাবে আপডেট করা হচ্ছে, সেভাবে করতে হবে –

প্রথম প্যারামিটার w_0 -এর জন্য,

$$w_0 = w_0 - \alpha \frac{\partial}{\partial w_0} J(W)$$

$$\frac{\partial}{\partial w_0} J(W) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_0^{(i)}$$

পরবর্তী প্যারামিটারগুলোর ($j = 1, 2, 3, \dots n$) জন্য,

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(W)$$

$$\frac{\partial}{\partial w_j} J(W) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_j^{(i)} - \frac{\lambda}{m} w_j$$

এই হলো একটি লিনিয়ার সাপোর্ট ভেক্টর মেশিন কীভাবে তৈরি করতে হয় তার বর্ণনা। পরবর্তী পরিচ্ছেদে আমরা দেখব কীভাবে, এত জটিল গাণিতিক হিসাবকিতাব একটু এড়িয়ে সহজ করে একটি লিনিয়ার সাপোর্ট ভেক্টর মেশিন তৈরি করা যায়।

পরিচ্ছেদ ৫.২ : লিনিয়ার সাপোর্ট ভেক্টর মেশিনের একটি সহজ উদাহরণ

আমরা এতক্ষণ যে উদাহরণটি ব্যবহার করলাম, সেটিই আমরা আবার ব্যবহার করব। আমরা ইতিমধ্যেই দেখেছি যে, উদাহরণে আমাদের সাপোর্ট ভেক্টর হচ্ছে তিনটি। এদেরকে S_1, S_2 ও S_3

নাম দিই। সেই সঙ্গে, Class 1-কে নেগেটিভ ক্লাস এবং Class 2-কে পজিটিভ ক্লাস বলে চিহ্নিত করি।

$$\text{সুতরাং } S_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, S_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \text{ এবং } S_3 = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

আমরা, এই তিনটি সাপোর্ট ভেস্টেরের প্রতিটির সঙ্গে '1' অগমেন্ট (Augment) করব (শেষে একটি করে রো যোগ করব এবং মান হবে 1) বায়াস (Bias) হিসেবে। এর গাণিতিক কিছু গুরুত্ব আছে, আমরা পরবর্তী সময়ে সেটি নিয়ে আরো বিষদভাবে আলোচনা করব। আপাতত, '1' অগমেন্ট করে নতুন সাপোর্ট ভেস্টেরগুলো হবে –

$$\tilde{S}_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, \tilde{S}_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \text{ ও } \tilde{S}_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

এখন, যেহেতু আমাদের সাপোর্ট ভেস্টের তিনটি, তাই আমাদের তিনটি প্যারামিটার থাকবে, এদেরকে আমরা α_1, α_2 ও α_3 দিয়ে চিহ্নিত করব, যারা যথাক্রমে \tilde{S}_1, \tilde{S}_2 ও \tilde{S}_3 -এর সঙ্গে সম্পৃক্ত।

এখন এই তিনটি প্যারামিটারের মান বের করার জন্য আমাদের তিনটি লিনিয়ার ইকুয়েশন সমাধান করতে হবে –

$$\alpha_1 \cdot \tilde{S}_1 \cdot \tilde{S}_1 + \alpha_2 \cdot \tilde{S}_2 \cdot \tilde{S}_1 + \alpha_3 \cdot \tilde{S}_3 \cdot \tilde{S}_1 = -1 \quad (\tilde{S}_1 \text{ Negative Class Point})$$

$$\alpha_1 \cdot \tilde{S}_1 \cdot \tilde{S}_2 + \alpha_2 \cdot \tilde{S}_2 \cdot \tilde{S}_2 + \alpha_3 \cdot \tilde{S}_3 \cdot \tilde{S}_2 = -1 \quad (\tilde{S}_2 \text{ Negative Class Point})$$

$$\alpha_1 \cdot \tilde{S}_1 \cdot \tilde{S}_3 + \alpha_2 \cdot \tilde{S}_2 \cdot \tilde{S}_3 + \alpha_3 \cdot \tilde{S}_3 \cdot \tilde{S}_3 = +1 \quad (\tilde{S}_3 \text{ Positive Class Point})$$

এই সমীকরণগুলো সমাধান করে আমাদেরকে তিনটি প্যারামিটার α_1, α_2 ও α_3 -এর মান বের করতে হবে। এখন সমীকরণগুলোতে \tilde{S}_1, \tilde{S}_2 ও \tilde{S}_3 -এর মান বসিয়ে পাই –

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_3 \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_2 \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} + \alpha_2 \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} + \alpha_3 \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = +1$$

সমীকরণগুলো সরল করে পাই –

অধ্যায় ৫ : সাপোর্ট ভেক্টর মেশিন (Support Vector Machine - SVM)

$$6\alpha_1 + 4\alpha_2 + 9\alpha_3 = -1$$

$$4\alpha_1 + 6\alpha_2 + 9\alpha_3 = -1$$

$$9\alpha_1 + 9\alpha_2 + 17\alpha_3 = +1$$

এখানে সরলীকরণের জন্য সাধারণ ভেক্টর ডট গুণন পদ্ধতি প্রয়োগ করা হয়েছে। আমি ধরে নিচ্ছি, সবাই উচ্চ মাধ্যমিক পর্যায়ে ভেক্টরের ডট গুণন পড়েছেন। আমি তা-ও এখানে আরেকবার সেটি লিখে দিচ্ছি –

ধরি, \vec{A} ও \vec{B} দুটি ভেক্টর এবং $\vec{A} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$ এবং $\vec{B} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$

তাহলে ভেক্টরদ্বয়ের ডট গুণন, $\vec{A} \cdot \vec{B} = a_1b_1 + a_2b_2 + a_3b_3$

সূতরাং, এখন ওপরের তিনটি সমীকরণ সমাধান করে পাই, $\alpha_1, \alpha_2 = -3.25$ এবং $\alpha_3 = 3.5$ ।

আমরা আমাদের সব প্যারামিটারের মান পেয়ে গিয়েছি। এখন, এগুলো ব্যবহার করে আমাদেরকে আমাদের ডিসিশন বাউন্ডারি হিসেবে ব্যবহৃত হাইপার প্লেন (Hyper Plane)-টির সমীকরণ বের করতে হবে। এটি বের করতে পারলেই আমাদের কাজ শেষ।

আমাদের হাইপার প্লেনের সমীকরণকে যদি $y = wx + b$ ধরি, যেখানে b হচ্ছে বায়াস এবং W হচ্ছে ওয়েইট ভেক্টর, ঠিক যেরকম আমরা লিনিয়ার রিগ্রেশন করার সময় দেখেছিলাম, তাহলে \tilde{W} হবে W -এর অগমেন্টেড রূপ।

এখন, ডিসিশন বাউন্ডারি হাইপার প্লেন বের করার জন্য আগে আমাদের \tilde{W} বের করে নিতে হবে, এর সূত্র হচ্ছে –

$$\tilde{W} = \sum_i \alpha_i S_i$$

এখন, এই সমীকরণে সব মান বসিয়ে পাই,

$$\tilde{W} = (-3.25) \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + (-3.25) \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + (3.5) \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix}$$

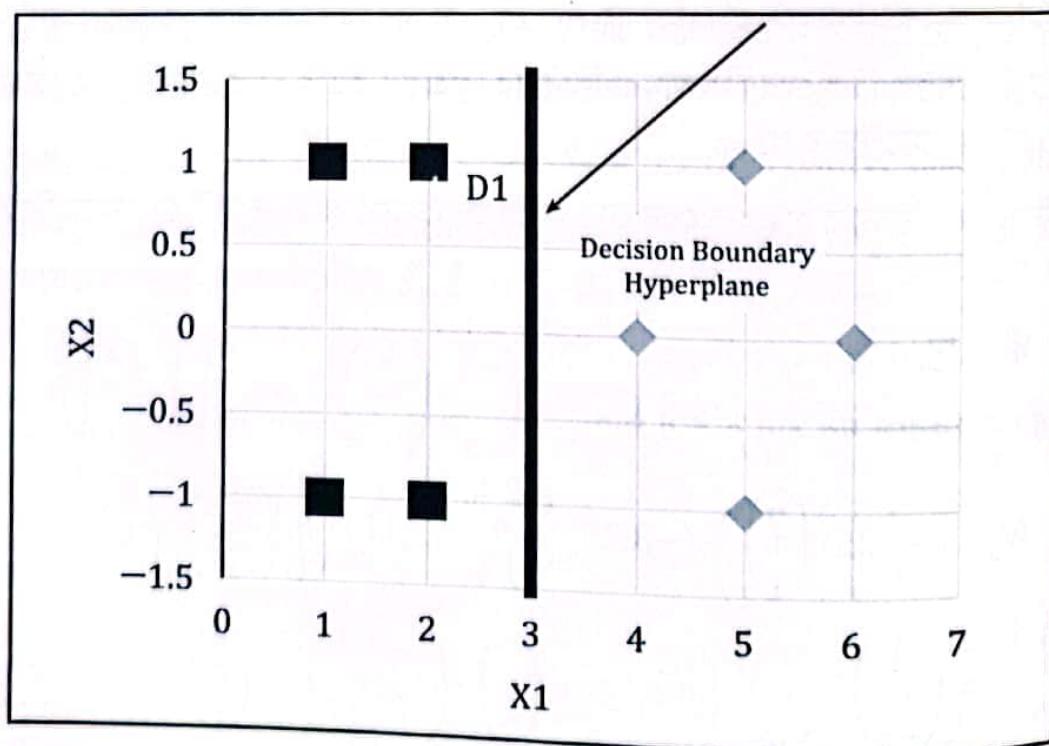
এখন, আমরা বলেছিলাম যে আমরা অগমেন্ট '1' ব্যবহার করছি বায়াস হিসেবে, যেটি একেবারে শেষের গো-এর অতিরিক্ত '1'-টিকে বোঝায়, \tilde{S}_1, \tilde{S}_2 ও \tilde{S}_3 -এর ক্ষেত্রে।

সুতরাং, সে হিসেবে ধরতে গোলে, $\tilde{W} = \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix}$ -এ -3 হচ্ছে বায়াস এবং $W = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ ।

দুটো বিষয় একটু মনে রাখতে হবে –

- বায়াসের মান ঝণাত্তক মানে অফসেট হবে ধনাত্তক, আর বায়াসের মান ধনাত্তক মানে অফসেট হবে ঝণাত্তক। অফসেট বলতে এখানে সরণ বোঝানো হচ্ছে। অর্থাৎ, ডিসিশন বাউন্ডারি কোনদিকে কতটুকু সরবে, সেটি।
- $W = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ মানে ভার্টিকাল বা উল্লম্বরেখা আর $W = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ মানে হরাইজন্টাল বা আনুভূমিক রেখা।

এখন এই W এবং বায়াসের মান $y = wx + b$ সমীকরণে বসিয়ে, আমরা পেয়ে যাব আমাদের কাঞ্জিক্ষত ডিসিশন বাউন্ডারি হাইপার প্লেন। এখানে লক্ষণীয় বিষয় যে, বায়াস যদি 3 হতো, তাহলে ডিসিশন বাউন্ডারি থাকত X -অক্ষের ঝণাত্তক দিকে, 3 ঘর সরে। আর যদি $W = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ হতো, তাহলে ডিসিশন বাউন্ডারি হতো X -অক্ষের সমান্তরাল। এ দুটো শুধুই কিছু শর্টকাট, সহজে অল্প সময়ে ডিসিশন বাউন্ডারি বের করার জন্য। ডিসিশন বাউন্ডারি যদি দুটোর একটিও না হয়, তাহলে আমাদের সাধারণভাবে সমীকরণে বসিয়ে সমাধান করতে হবে।



গ্রাফ 5.2.1

শেষমেশ বাকি থাকে, নতুন পয়েন্টকে ক্লাসিফাই করা।

অধ্যায় ৫ : সাপোর্ট ভেস্টার মেশিন (Support Vector Machine - SVM)

ধৰা যাক, নতুন একটি পয়েন্ট $x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

আমাদের ডিসিশন বাউন্ডারি, $y = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} - 3 = 1 - 3 = -2$

এখন, যেহেতু, $y \leq 0$ সূতৰাং এটি অবশ্যই নেগেটিভ ক্লাস অর্থাৎ Class 1-এর অন্তর্ভুক্ত, যেটি আমরা শুরুতেই বলে নিয়েছি। যদি, $y > 0$ হতো, তবে এটি পজিটিভ ক্লাস অর্থাৎ Class 2-এর অন্তর্ভুক্ত হতো।

তাহলে এই হলো লিনিয়ার সাপোর্ট ভেস্টার মেশিন বা LSVM। আশা করি, সবাই বুঝতে পেরেছেন। এই উদাহরণটিতে একটু ভিন্নভাবে সাপোর্ট ভেস্টার মেশিনের ধারণা বোঝানো হয়েছে। এরপরে আমরা দেখব সাপোর্ট ভেস্টার মেশিন কীভাবে নন-লিনিয়ার ডেটার জন্য কাজ করে।

পরিচেদ ৫.৩ : কার্নেল ট্রিক (Kernel Trick) ও নন-লিনিয়ার এসডিএম (Non-Linear SVM)

আমরা এতক্ষণ যে ডেটা নিয়ে কাজ করলাম, সেগুলো ছিল লিনিয়ারলি সেপারেবল (Linearly Separable) ডেটা। সহজ করে বলতে গেলে, এ ধরনের ডেটার বৈশিষ্ট্য হচ্ছে, এদেরকে একটি রেখা টেনে আলাদা করে ফেলা যায়। যেরকম, আমাদের আগের ব্যবহৃত ডেটাসেটটি (গ্রাফ 5.3.1 - মোটা দাগ টেনে দিয়ে আলাদা করে ফেলা হয়েছে দুই ক্লাসের ডেটাকে)।

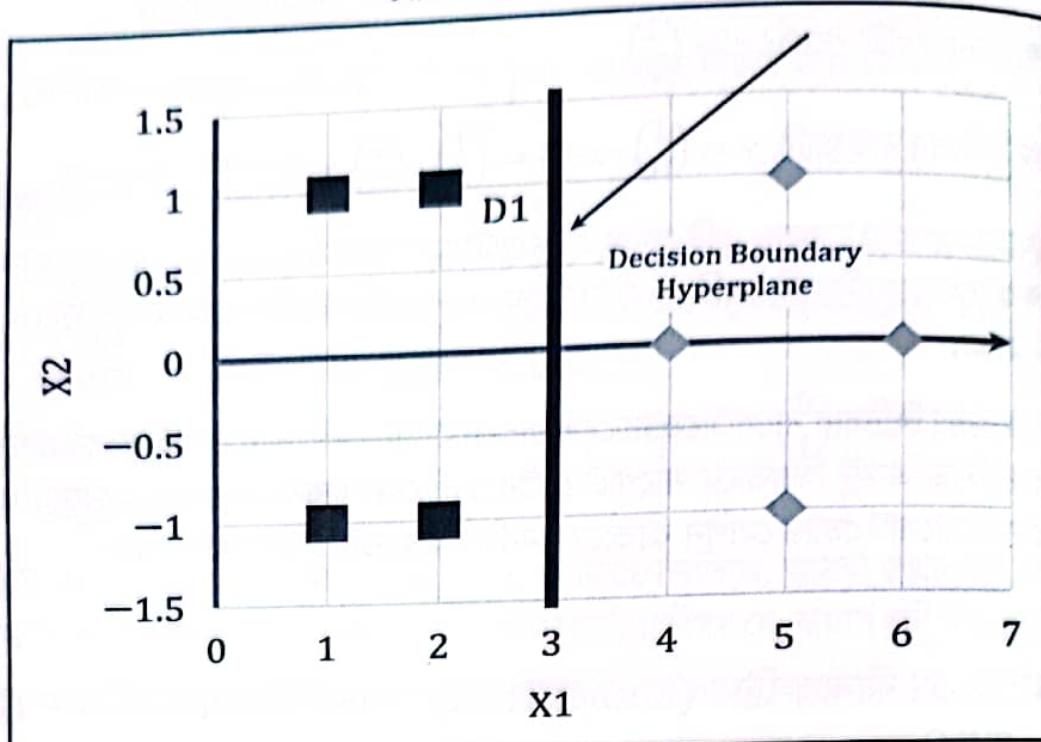
এখন, যদি ডেটাসেট এত সোজা না হয়? যদি, ডেটাসেটের ডেটা একটিমাত্র লিনিয়ার ডিসিশন বাউন্ডারি দিয়ে আলাদা করা না যায়? গ্রাফ 5.3.2 দেখুন।

যদি ডেটাসেট হয় এরকম? তাহলে কীভাবে আলাদা করা যাবে দুটি ক্লাসের ডেটাকে? একটিমাত্র সোজা দাগ টেনে তো এটি সম্ভব নয়। তাহলে?

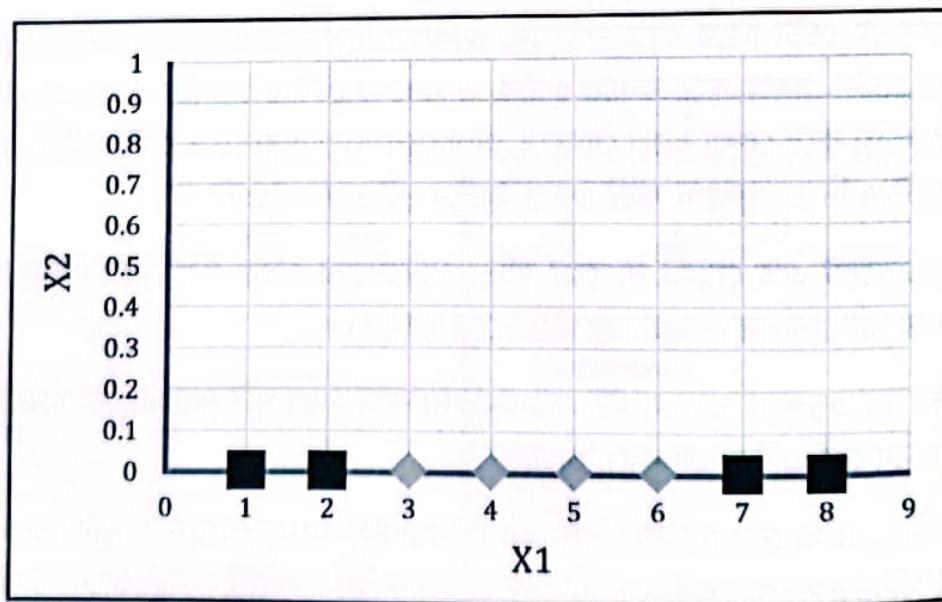
প্রথমে বলে নিই, এ ধরনের ডেটাকে বলা হয় লিনিয়ারলি নন-সেপারেবল ডেটা। এর অর্থ হচ্ছে, এদেরকে একটিমাত্র সরলরেখা দিয়ে আলাদা করা যাবে না। এজন্য, অন্য কোনো উপায় অবলম্বন করতে হবে।

এই অন্য উপায়ের একটি গালভরা নাম আছে – ‘কার্নেল ট্রিক’ (Kernel Trick)।

কার্নেল ট্রিক ব্যবহার করে, এরকম লিনিয়ারলি নন-সেপারেবল ডেটাকে আমরা SVM-এর সাহায্যে ক্লাসিফাই করতে পারি।



গ্রাফ 5.3.1



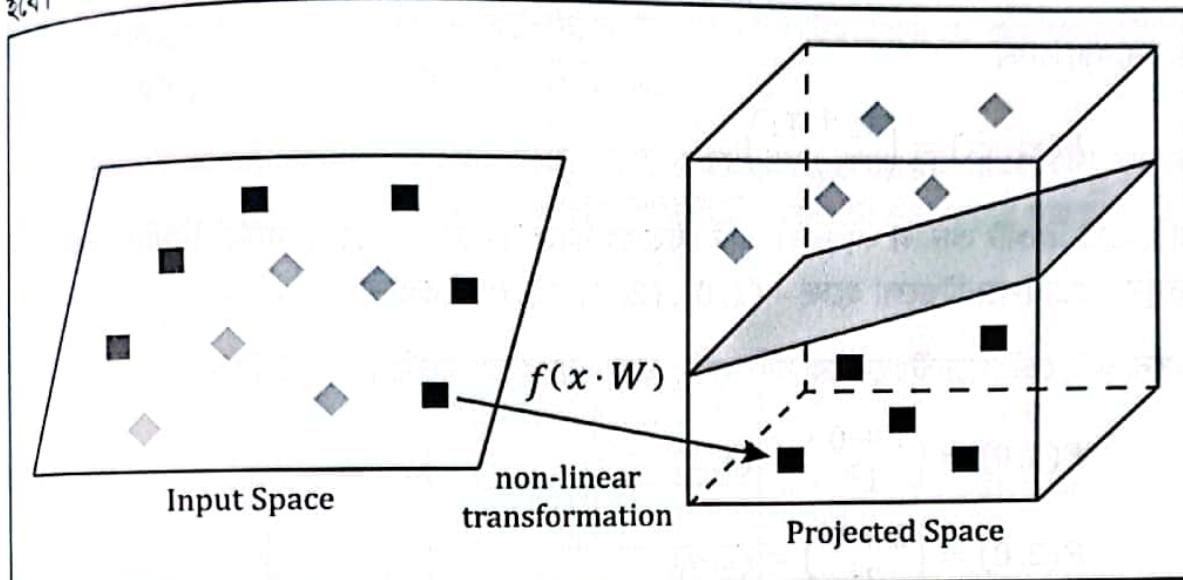
ছবি 5.3.2

কার্নাল ট্রিক ব্যাপারটি একটু নয়, বেশ জটিল। এর ভেতরকার থিওরি দেখাতে হলে অনেক গভীর গিয়ে অনেকগুলো গাণিতিক জটিল তত্ত্ব বোঝাতে হবে, যেটি আমাদের এই বইয়ের আওতার বাইরে বলা যায়। তাই আমরা অস্তত এই বইয়ের জন্য সেসব জটিল গাণিতিক হিসাবনিকাশ বাদ দিয়ে খুব সহজভাবে আসল বিষয়টি কী ঘটছে সেটি বোঝার চেষ্টা করব। আমাদের মূল লক্ষ্য হবে

অধ্যায় ৫ : সাপোর্ট ভেক্টর মেশিন (Support Vector Machine - SVM)

শুধু বিষয়টি সহজ করে বুঝে রাখা, যাতে সেটি আমরা পাইথনের মেশিন লার্নিং লাইব্রেরি ব্যবহার করার সময় বুঝতে পারি যে কী করছি।

যাই হোক, প্রথমেই বললাম যে আমাদের ডেটা এবার হচ্ছে লিনিয়ার নন-সেপারেবল ডেটা। এদেরকে আলাদা করার জন্য আমাদেরকে কার্নাল ট্রিক নামে এক ধরনের উপায় অবলম্বন করতে হবে।



ছবি 5.3.3

ছবি 5.3.3 দেখলে ধারণা পাবেন যে, প্রকৃতপক্ষে আমরা কী করতে চাইছি। ওপরে ছবিতে, দেখুন, আমরা যদি ইনপুট স্পেস (Input Space)-এর দিকে তাকাই (যে ডেটা আমরা ইনপুট দিচ্ছি), তাহলে দেখব যে লাল ও নীল এই দুই ক্লাসের বিন্দুকে আমরা শুধু একটি সরলরেখা টেনে আলাদা করতে পারব না। একটু লক্ষ করবেন, আমাদের ইনপুট স্পেস এখানে দ্বিমাত্রিক বা 2D, তাই আমরা একে একমাত্রিক বা 1D সরলরেখা দিয়ে আলাদা করার কথা বলেছি। যদি ত্রিমাত্রিক বা 3D ডেটা হতো, তবে তাদেরকে একটি দ্বিমাত্রিক বা 2D প্লেন এঁকে আলাদা করার কথা বলতাম, যেটি ছবিতে প্রোজেক্টেড স্পেস (Projected Space)-এ দেখানো হয়েছে।

এখন, আমরা এখানে একটি ফাংশন $f()$ ব্যবহার করেছি, যার মধ্যে আমরা আমাদের ইনপুট স্পেসের ডেটা ইনপুট দিয়েছি এবং এই ফাংশনটি সেই 2D ইনপুট ডেটাকে পরিবর্তন করে, ডেটাকে উচ্চতর ডাইমেনশনে নিয়ে গিয়ে 2D ডেটাকে 3D ডেটায় পরিবর্তন করেছে। ডেটাকে 2D থেকে 3D-তে পরিবর্তন করার পরে (মনে করুন, মেঝেতে অনেকগুলো বল পড়ে আছে, অর্থাৎ, বলগুলো দ্বিমাত্রিক অবস্থানে আছে; আমরা সবগুলো বল নিয়ে বিভিন্ন বেগে ওপরের দিকে ছুঁড়ে দিয়েছি, এখন তারা একটি ত্রিমাত্রিক অবস্থানে আছে) দেখা গেল, এখন আমরা, খুব সহজেই একটি 2D প্লেন (সবুজ রঙের) দিয়ে লাল এবং নীল ক্লাসের ডেটাকে আলাদা করতে পারছি। এখানে 2D

থেকে 3D-তে নিয়ে যাওয়ার জন্য যে ম্যাজিক ফাংশনটি আমরা ব্যবহার করলাম, সেটিই হচ্ছে কার্নাল ফাংশন।

কার্নাল ফাংশন, সহজ করে বললে এক ধরনের ম্যাপিং ফাংশন। সেই ফাংশনের ভেতরে আমরা কিছু সংখ্যা ইনপুট দেব, কার্নাল ফাংশন সেই ইনপুট ডেটা নিয়ে কিছু গুণ-ভাগ করে আমাদের নতুন কতগুলো সংখ্যা ফেরত দেবে।

যেমন ধরা যাক,

$$F(x_1, x_2) = \begin{pmatrix} x_1 + x_2 \\ x_1^2 \end{pmatrix}$$

ধরি, এটি একটি কার্নাল ফাংশন। এটি যেভাবে কাজ করবে, আমাদের নন-লিনিয়ার গ্রাফ থেকে দেখুন – ডেটাপয়েন্টগুলো হচ্ছে – $(1, 0), (2, 0), (3, 0)$ ইত্যাদি।

এখন, এই ডেটাপয়েন্টগুলোকে যদি আমি এক এক করে কার্নেলে ইনপুট দিই –

$$F(1, 0) = \begin{pmatrix} 1 + 0 \\ 1^2 \end{pmatrix} = (1, 1)$$

$$F(2, 0) = \begin{pmatrix} 2 + 0 \\ 2^2 \end{pmatrix} = (2, 4)$$

$$F(3, 0) = \begin{pmatrix} 3 + 0 \\ 3^2 \end{pmatrix} = (3, 9) \text{ ইত্যাদি।}$$

তাহলে, এইগুলো হবে আমাদের নতুন ফিচার ডেটা পয়েন্ট। এই নতুন ফিচার ডেটা পয়েন্ট নিয়ে আমরা কাজ করব। আমাদের লক্ষ্য হবে এমন একটি কার্নাল ফাংশন খুঁজে বের করা, যেটি ব্যবহার করে আমরা যে নতুন ফিচার ডেটা পয়েন্টগুলো পাব সেগুলো যদি গ্রাফে প্লট করি তাহলে দেখব যে আমাদের নতুন ফিচার ডেটা এমনভাবে প্লট হয়েছে, যাতে তাদের আমরা লিনিয়ারলি সেপারেট করতে পারি (একটি সরলরেখা টেনে, কিংবা একটি হাইপার প্লেন দিয়ে)।

সাধারণত কার্নাল ফাংশনে আরেকটি কাজ করা হয়, সেটি হচ্ছে, আমাদের মূল ইনপুট ডেটার যে ডাইমেনশন, কার্নাল ফাংশন ব্যবহার করে ফিচার তৈরি করে নেওয়ার সময় সেই ডেটার ডাইমেনশন বাড়িয়ে দেওয়া হয়।

যেরকম ধরা যাক,

$$F(x_1, x_2) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ x_1^2 + x_2^2 \end{pmatrix}$$

অধ্যায় ৫ : সাপোর্ট ভেক্টর মেশিন (Support Vector Machine - SVM)

এখন দেখুন, আমাদের ইনপুট ডেটার ডাইমেনশন ছিল 2, কিন্তু নতুন পরিবর্তিত ডেটার ডাইমেনশন 3; ঠিক যেরকম আমরা শুরুর উদাহরণে দেখলাম। এই নতুন হায়ার-ডাইমেনশনাল ডেটাই আমাদের এখন ফিচার হিসেবে ব্যবহৃত হবে।

এখন প্রশ্ন হচ্ছে, আমরা কীভাবে বুবুব কোন ফাংশন কার্নাল হিসেবে ব্যবহার করতে হবে? সাধারণত, সাপোর্ট ভেক্টর মেশিনের ক্ষেত্রে কিছু জনপ্রিয় কার্নাল আছে, সেগুলোই সাধারণ ক্ষেত্রে ঘূরেফিরে ব্যবহার করা হয়। আর ক্ষেত্রবিশেষে, নতুন আরো অপটিমাইজড কোনো কার্নাল তৈরি করে নেওয়া হয় ডেটাসেটের ধরনের ওপরে ভিত্তি করে।

আমরা প্রথমেই একটি কার্নাল ফাংশন নিয়ে কাজ করব, একে বলা হয় গাউসিয়ান কার্নাল (Gaussian Kernel)। এই কার্নাল ব্যবহার করে, প্রতিটি ফিচার ডেটা বের করার সূত্র হচ্ছে –

$$f_i = e^{-\frac{\|x-x_i\|^2}{2\sigma^2}}$$

x_1	x_2
2	3
3	4
4	5

টেবিল 5.5.1

ধরা যাক, টেবিল 5.5.1 আমাদের ডেটাসেট। এখন, আমরা এই ডেটাসেটের (2, 3) পয়েন্টটির জন্য গাউসিয়ান কার্নাল ব্যবহার করে নতুন ফিচার পয়েন্ট বের করব। আমাদের এই পয়েন্টের ইনপুট ডেটাতে ফিচার আছে দুটি, $x_1 = 2$ ও $x_2 = 3$ ।

এখন, আমরা যদি এই ডেটা পয়েন্টকে আমাদের গাউসিয়ান কার্নালে ইনপুট হিসেবে দিই, তাহলে আমাদের নতুন 3টি ফিচার তৈরি হবে, যেহেতু আমাদের ট্রেনিং ডেটার সংখ্যা 3টি। নতুন ফিচারগুলো হলো –

$$f_1 = e^{-\frac{\|x-x_1\|^2}{2\sigma^2}} = e^{-\frac{(2-2)^2+(3-3)^2}{2\sigma^2}} \approx 1$$

$$f_2 = e^{-\frac{\|x-x_2\|^2}{2\sigma^2}} = e^{-\frac{(2-3)^2+(3-4)^2}{2\sigma^2}} \approx 0$$

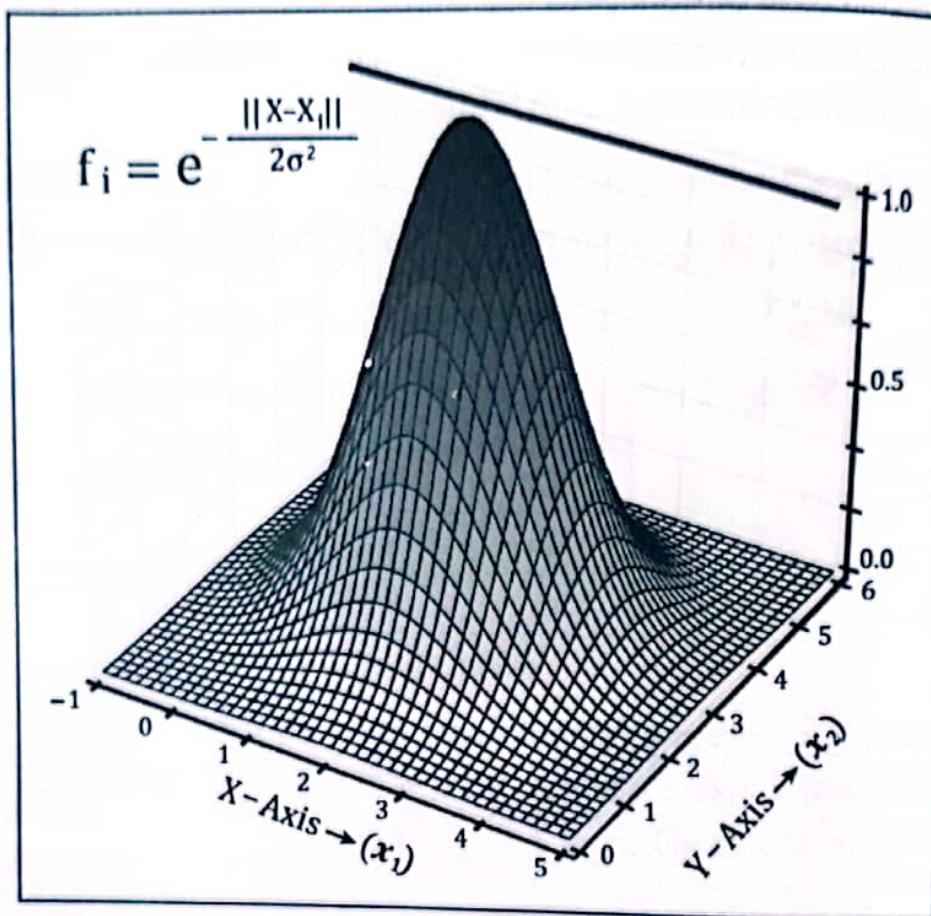
$$f_3 = e^{-\frac{\|x-x_3\|^2}{2\sigma^2}} = e^{-\frac{(2-4)^2+(3-5)^2}{2\sigma^2}} \approx 0$$

এই হিসাবগুলোতে σ^2 -এর একটি মান আমরা ধরে নিয়েছি 1। সুতরাং আমরা নতুন তিনটি ফিচার ভ্যালু পেয়ে গেলাম,

$$(f_1, f_2, f_3) \approx (1, 0, 0)$$

এরপর থেকে $(2, 3)$ পয়েন্টটির বদলে এর নতুন ফিচার ডেটার $f = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ নিয়েই আমাদের কাজ

করতে হবে। নিচের ছবিটি দেখি (ছবি 5.3.4) :



ছবি 5.3.4

গ্রাফটিতে দেখা যাচ্ছে, ফিচার-এর মান '1' কেবল তখনই হয়, যখন আমরা X-অক্ষ ও Y-অক্ষ বরাবর যথাক্রমে 2 ও 3 একক নিচ্ছি। বাকি ডেটা পয়েন্ট দুটির ক্ষেত্রে ফিচারের মান শূন্যের কাছাকাছি।

এ থেকে বোঝা যাচ্ছে, আমাদের এই কার্নালটি আসলে যা করছে তা হলো, এটি যে পয়েন্ট ইনপুট হিসেবে নিচ্ছে, তার সঙ্গে ট্রেনিং ডেটার সব পয়েন্টের সামঞ্জস্য (Similarity) বের করছে। যখন ট্রেনিং ডেটার মধ্যেকার পয়েন্ট $(2,3)$ -এর সঙ্গে, অর্থাৎ নিজের সঙ্গে সামঞ্জস্য খুঁজে বের করছে তখনই কেবল ফিচারের মান '1' হচ্ছে, অন্যান্য ক্ষেত্রে শূন্যের কাছাকাছি হচ্ছে।

এখন ধরে নিই, আমাদের হাইপোথিসিস আগের মতোই –

অধ্যায় ৫ : সাপোর্ট ভেক্টর মেশিন (Support Vector Machine - SVM)

$$h_w(x) = \begin{cases} 0, & W^T x < 0 \\ 1, & W^T x \geq 0 \end{cases}$$

গুণে পরিবর্তন আসবে, তা হলো x (ইনপুট-এর সময় যা ফিচার হিসেবে ধরেছিলাম)-এর বদলে আমরা f (নতুন ফিচার ভেক্টর) নেব।

তাহলে, আমাদের নতুন হাইপোথিসিস দাঁড়াবে –

$$h_w(x) = \begin{cases} 0, & W^T f < 0 \\ 1, & W^T f \geq 0 \end{cases}$$

বাকি থাকল, আমরা W অর্থাৎ আমাদের প্যারামিটার সেট কীভাবে পাব এবং আমাদের অপটিমাইজেশন ফাংশন কী হবে? প্যারামিটার সেট পাব ঠিক আগের নিয়মে –

প্রথম প্যারামিটার w_0 -এর জন্য,

$$w_0 = w_0 - \alpha \frac{\partial}{\partial w_0} J(W)$$

$$\frac{\partial}{\partial w_0} J(W) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_0^{(i)}$$

প্রবর্তী প্যারামিটার ($j = 1, 2, 3, \dots n$)-এর জন্য,

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(W)$$

$$\frac{\partial}{\partial w_j} J(W) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_j^{(i)} - \frac{\lambda}{m} w_j$$

আর অপটিমাইজেশন ফাংশন আমাদের LSVM-এর ক্ষেত্রে ছিল –

$$\min_w C \left[\sum_{i=1}^m \{y^{(i)} (\text{Cost}_1(z)) + (1 - y^{(i)}) (\text{Cost}_0(z))\} \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

যেখানে, $z = W^T x$ । এখন, x -এর বদলে শুধু f ব্যবহার করব, যেহেতু এটি আমাদের নতুন ফিচার ভেক্টর। সুতরাং, নন-লিনিয়ার সাপোর্ট ভেক্টর মেশিনের জন্য আমাদের নতুন অপটিমাইজেশন ফাংশন দাঁড়ায় –

$$\min_w C \left[\sum_{i=1}^m \left\{ y^{(i)} \left(\text{Cost}_1(W^T f^{(i)}) \right) + (1 - y^{(i)}) \left(\text{Cost}_0(W^T f^{(i)}) \right) \right\} \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

সবশেষে আরো কয়েকটি বিষয় বিবেচনা করতে হবে :

1. $C (= \frac{1}{\lambda})$ -এর মান নেওয়ার ক্ষেত্রে, যদি C -এর মান অনেক বড়ো নিই, তাহলে আমরা Low Bias, High Variance দেখতে পাব, অর্থাৎ ওভারফিটিং হওয়ার সম্ভাবনা থাকবে বেশি। আর যদি C -এর মান অনেক ছোটো নিই, তাহলে আমরা High Bias, Low Variance দেখতে পাব, অর্থাৎ আন্ডারফিটিং হওয়ার সম্ভাবনা থাকবে বেশি।
2. σ^2 -এর মান অনেক বড়ো হলে, High Bias, Low Variance, আর মান অনেক ছোটো হলে Low Bias, High Variance। এর কারণ হচ্ছে, σ^2 -এর মান অনেক বড়ো হলে ফিচার খুব Smoothly Vary করতে পারবে, আর ছোটো হলে তা পারবে না।
3. গাউসিয়ান কার্নালের মতো আরো কিছু জনপ্রিয় কার্নাল আছে –
 - Polynomial Kernel:

$$K(x_1, x_2) = (x_1 \cdot x_2 + 1)^d \quad [d = \text{degree of polynomial}]$$
 - Gaussian Radial Basis Function (RBF):

$$K(x_1, x_2) = e^{-\gamma |x_1 - x_2|^2} \quad [\text{for } \gamma > 0, \text{ mostly } \gamma = \frac{1}{2\sigma^2}]$$
 - Laplace RBF Kernel:

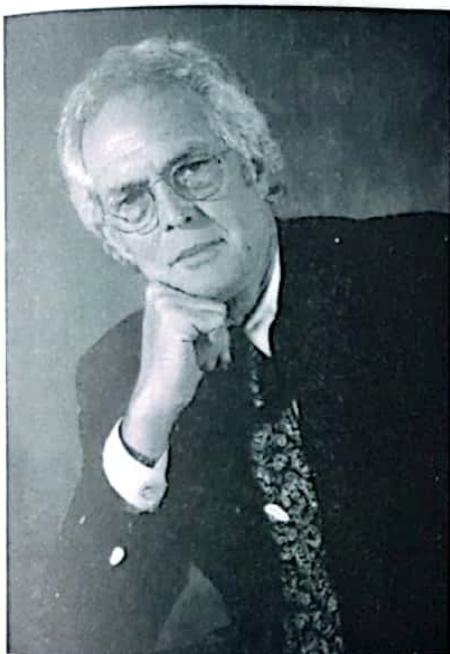
$$K(x_1, x_2) = e^{-\frac{|x_1 - x_2|}{\sigma}}$$
 - Linear Kernel (No Kernel at all):

$$h_w(x) = \begin{cases} 0, & W^T x < 0 \\ 1, & W^T x \geq 0 \end{cases}$$

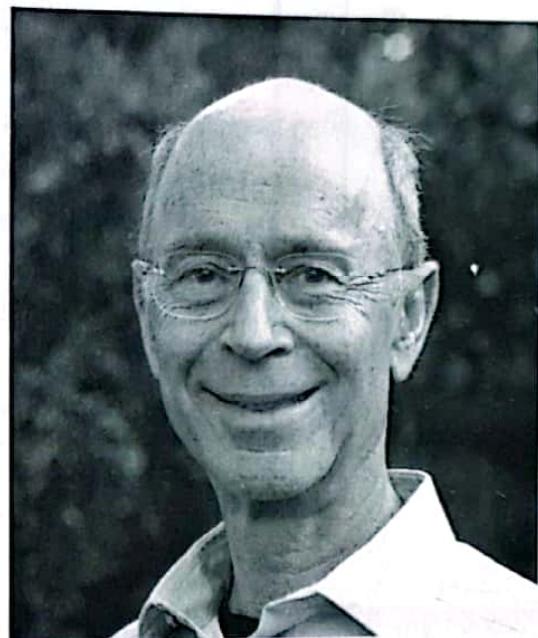
এই ছিল আমাদের সাপোর্ট ভেস্ট্র-সম্পর্কিত যাবতীয় আলোচনা।

অধ্যায় ৬: কে-নিয়ারেস্ট নেইবরস (K-Nearest Neighbors)

কে-নিয়ারেস্ট নেইবরস (K-Nearest Neighbors) বা KNN হচ্ছে একটি জনপ্রিয় এবং খুবই সহজ ক্লাসিফিকেশন অ্যালগরিদম। এটি প্রথম ব্যবহার করেন ফিক্স (Evelyn Fix, 1904 - 1965) ও হেজেস (Joseph Lawson Hodges Jr., 1922 - 2000) নামের দুজন বিজ্ঞানী 1951 সালে। পরবর্তী সময়ে কভার (Thomas M. Cover, 1938 - 2012) ও হার্ট (Peter E. Hart, 1941) নামে দুজন বিজ্ঞানী এটি উন্নতকরণের কাজ করেন।



Tomas M. Cover (1938 - 2012)



Peter E. Hart (1941)

ছবি 6.1

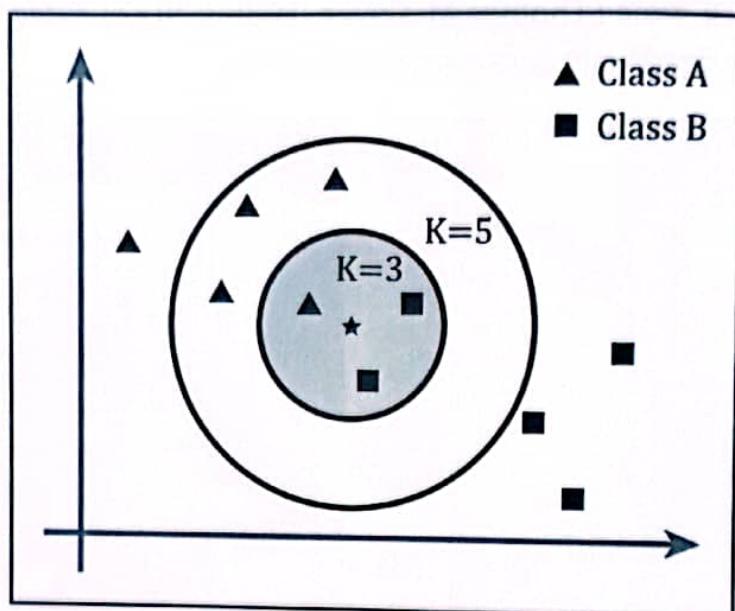
আমি যখন আমার ছাত্রছাত্রীদের মেশিন লার্নিং পড়াই, তারা সবচেয়ে সহজে বুঝতে পারে এই অ্যালগরিদমটি। এটি খুবই সহজ একটি অ্যালগরিদম, কিন্তু এর বাস্তবিক প্রয়োগ (যদি সঠিকভাবে করা যায়) হতে পারে খুবই ফলপ্রসূ। সোজা কথায় যদি বোঝাতে চাই, এটি একটি ভোট গণনা করার মতো অ্যালগরিদম।

ধ্রীয়া যাক, আপনি কাচি বিরিয়ানি খেতে চাইছেন হট করে। আপনি আপনার তিনজন বন্ধুকে জিজ্ঞাসা করলেন কোথায় কাচি বিরিয়ানি ভালো হবে? পুরান ঢাকার কোনো দোকানে, নাকি স্টার কাবাবের কাচি? দুজন বন্ধু পুরান ঢাকার পক্ষে মতামত দিল, আর একজন দিল স্টারের পক্ষে। এখন তাহলে আপনি কোনটায় যাবেন? যেটায় বেশি মানুষ আপনাকে যেতে বলছে, অর্থাৎ পুরান

ঢাকাতে, সেটাতেই তো যাবেন, তাই নয় কি? যদি এটি বুঝে থাকেন, তাহলে এই অ্যালগরিদম আপনি বুঝে গেছেন অনেকখানিই।

পরিচেদ ৬.১ : KNN-এর সাধারণ ধারণা

নিচের ছবিটি দিয়ে বোঝা শুরু করি :



ছবি 6.1.1

ছবিতে দেখুন, দুটি ভিন্ন ভিন্ন আকৃতির ডেটা পয়েন্ট আছে, ত্রিভুজ ও বর্গ। ধরা যাক, এই দুই আকৃতি দিয়ে দুটি ভিন্ন ক্লাস বোঝানো হয়েছে (মনে করি ত্রিভুজাকৃতি দিয়ে কুকুরের ছবি, আর বর্গাকৃতি দিয়ে বিড়ালের ছবি)। ছবি অনুযায়ী, প্রতিটি ডেটার জন্য দুটি করে ফিচার নেওয়া হয়েছে, x_1 ও x_2 । মাঝখানে একটি তারকা চিহ্ন আঁকা আছে, এটি হচ্ছে আমাদের অজানা ডেটা পয়েন্ট, অর্থাৎ আমরা জানি না, এটি কুকুর নাকি বিড়ালের ছবি। এখন, আমাদের যেটি করতে হবে, সেটি হচ্ছে, এই অজানা ডেটা পয়েন্টটিকে ক্লাসিফাই করতে হবে, অর্থাৎ বলতে হবে এটি কুকুরের ছবি নাকি বিড়ালের ছবি।

কীভাবে পুরো কাজটি করব, সেটি বিস্তারিত আলোচনার আগে আমি খুব সংক্ষেপে বলে দিই, কী করব আমরা। প্রথমে, আমরা K-এর একটি মান ধরে নেব। সাধারণত K-এর মান বেজোড় সংখ্যা ধরা হয় (1, 3, 5, 7... ইত্যাদি)। এর কারণ হচ্ছে, ধরুন যদি, $K = 4$ নিই, অর্থাৎ চারজন মানুষ ভোট দিচ্ছে। যদি দুজন ভোট দেয় যে নতুন ডেটা পয়েন্টটি কুকুরের ছবি, আর বাকি দুজন ভোট দেয় যে নতুন ডেটা পয়েন্টটি বিড়ালের ছবি, তাহলে সমান সমান ভোট হয়ে গেল না? তখন তো

অধ্যায় ৬ : কে-নিয়ারেস্ট নেইবৰস (K-Nearest Neighbors)

আমরা মুশকিলে পড়ে যাব – নতুন পয়েন্টটিকে তখন আমরা কী হিসেবে মানব? কুকুর, নাকি বিড়াল? এই মুসিবত থেকে যাতে আমরা বিরত থাকতে পারি, তাই K-এর মান সাধারণত বেজোড় সংখ্যা দেওয়া হয়।

এপরে আমাদের অজানা ডেটা পয়েন্টের আশপাশের সব জানা ডেটা পয়েন্ট থেকে সবচেয়ে কাছের K-সংখ্যক ডেটা পয়েন্ট আমরা বিবেচনা করব আমাদের পরবর্তী ধাপের জন্য। ওপরের চিত্রে দেখুন, $K = 3$ -এর জন্য ভেতরের ছোটো বৃত্ত এবং $K = 5$ -এর জন্য বাইরের বড়ো বৃত্তটি অঙ্কা হয়েছে। বড়ো বৃত্তের ভেতরে একটি বিন্দু অতিরিক্ত আছে, ওটি নিয়ে আপাতত মাথা ঘমানার দরকার নেই।

এখন ছোটো বৃত্তের ভেতরে দেখুন তিনটি ডেটা পয়েন্ট রয়েছে, যেগুলোর মধ্যে দুটি বর্গ ও একটি ত্রিভুজ। সুতরাং বর্গের সংখ্যা বেশি, অর্থাৎ বিড়ালের সংখ্যা বেশি। এর মানে হচ্ছে $K = 3$ নিয়ে আমরা দেখতে পাই যে অজানা ডেটা পয়েন্টটির সঙ্গে বিড়ালের সামঞ্জস্য বেশি, তাই এর আকৃতি তারকা থেকে বর্গ করে দিয়ে একে বিড়াল বলে বিবেচনা করা হবে।

একইভাবে, বড়ো বৃত্তের ভেতরে দেখুন ছয়টি ডেটা পয়েন্ট রয়েছে, যার মধ্যে চারটি ত্রিভুজ এবং দুটি বর্গ। সুতরাং ত্রিভুজের সংখ্যা বেশি অর্থাৎ কুকুরের সংখ্যা বেশি। এর মানে হচ্ছে $K = 5$ নিয়ে আমরা দেখতে পাই যে অজানা ডেটা পয়েন্টটির সঙ্গে কুকুরের সামঞ্জস্য বেশি, তাই এটির আকৃতি তারকা থেকে ত্রিভুজ করে দিয়ে একে কুকুর বলে বিবেচনা করা হবে। এভাবেই মূলত KNN অ্যালগরিদম কাজ করে।

তাহলে এর মূল ধাপ চারটি –

- অজানা ডেটা পয়েন্ট থেকে বাকি সব ডেটা পয়েন্টের দূরত্ব বের করতে হবে।
- দূরত্বের মান অনুযায়ী ছোটো থেকে বড়ো আকার (বা, Ascending Order)-এ ডেটা পয়েন্টগুলো সর্ট (sort) করে নিতে হবে।
- সর্ট করা ডেটা পয়েন্ট থেকে প্রথম K-সংখ্যক পয়েন্ট নিতে হবে।
- এই K-সংখ্যক ডেটা পয়েন্টের মধ্যে যে ক্লাসের পয়েন্ট সবচেয়ে বেশি সংখ্যক বার আছে, অজানা ডেটা পয়েন্টটিকে সেই ক্লাসে হিসেবে চিহ্নিত করতে হবে।

পরিচ্ছেদ ৬.২ : উদাহরণ

এখন আমরা একটি উদাহরণ দিয়ে পুরো অ্যালগরিদমটির বিস্তারিত বুবাব। নিচের চার্টটি দেখি (টেবিল 6.2.1)। এই চার্টে ওপরের চিত্রের ডেটা পয়েন্টগুলোর (ত্রিভুজ ও বর্গ) x_1 এবং x_2 ক্ষিতির ভ্যালুগুলোর মান দেওয়া আছে। সেই সঙ্গে আমরা কোন ডেটা পয়েন্ট কোন ক্লাসের

অন্তর্ভুক্ত, সেটিও শেষ কলামে লিখে দিয়েছি। এখানে 1 মানে কুকুর (ত্রিভুজ), 0 মানে বিড়াল (বর্গ)।

X_1	X_2	Class
4.2	2.8	1
4.0	2.0	1
3.8	0.5	1
2.0	1.5	1
2.7	2.5	1
1.7	3.2	0
2.7	4.0	0
1.2	5.2	0
2.2	6.2	0
0.3	6.2	0

টেবিল 6.2.1

অজানা ডেটা পয়েন্টের ফিচারে দুটির মান হচ্ছে (2.2, 3)। আমাদের এখন বের করতে হবে যে এই অজানা ডেটা পয়েন্ট কুকুর হবে, নাকি বিড়াল।

সেজন্য আমরা এখন যেটি করব, সবার প্রথমে (2.2, 3) পয়েন্ট থেকে টেবিলের সব ডেটা পয়েন্টের মধ্যেকার দূরত্ব বের করব। দূরত্ব বের করার জন্য আমরা ইউক্লিডীয় দূরত্বের সূত্র প্রয়োগ করব। দুটি পয়েন্ট (x_1, y_1) ও (x_2, y_2) -এর মধ্যেকার ইউক্লিডীয় দূরত্ব হচ্ছে,

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

এখন তাহলে আমরা সব ডেটা পয়েন্ট থেকে (2.2, 3) পয়েন্টের ইউক্লিডীয় দূরত্ব বের করে ফেলি:

X_1	X_2	Distance Formula	Distance Value	Sort Rank
4.2	2.8	$\sqrt{(4.2 - 2.2)^2 + (2.8 - 3)^2}$	2.00998	5
4.0	2.0	$\sqrt{(4 - 2.2)^2 + (2 - 3)^2}$	2.05913	6
3.8	0.5	$\sqrt{(3.8 - 2.2)^2 + (0.5 - 3)^2}$	2.96816	8

অধ্যায় ৬ : কে-নিয়ারেস্ট নেইবরস (K-Nearest Neighbors)

2.0	1.5	$\sqrt{(2 - 2.2)^2 + (1.5 - 3)^2}$	1.51327	4
2.7	2.5	$\sqrt{(2.7 - 2.2)^2 + (2.5 - 3)^2}$	0.70711	2
1.7	3.2	$\sqrt{(1.7 - 2.2)^2 + (3.2 - 3)^2}$	0.53852	1
2.7	4.0	$\sqrt{(2.7 - 2.2)^2 + (4 - 3)^2}$	1.11803	3
1.2	5.2	$\sqrt{(1.2 - 2.2)^2 + (5.2 - 3)^2}$	2.41661	7
2.2	6.2	$\sqrt{(2.2 - 2.2)^2 + (6.2 - 3)^2}$	3.2	9
0.3	6.2	$\sqrt{(0.3 - 2.2)^2 + (6.2 - 3)^2}$	3.72156	10

টেবিল 6.2.2

এখন আমরা এই ডেটা পয়েন্টগুলোকে দূরত্বের মানে ছোটো থেকে বড়ো ক্রম বা অ্যাসেন্ডিং অর্ডার (ascending order)-এ সর্ট করি। সর্ট করার পরে টেবিলে ডেটা পয়েন্টগুলোর অবস্থান গেরের টেবিলে Sort Rank হিসেবে দেওয়া আছে। সেখান থেকে ডেটা পয়েন্টগুলো নিয়ে যদি আমরা একটি আলাদা টেবিলে একই ক্রমে সাজাই, তাহলে নিচের টেবিলের মতো দাঁড়াবে :

X_1	X_2	Class
1.7	3.2	0
2.7	2.5	1
2.7	4.0	0
2.0	1.5	1
4.2	2.8	1
4.0	2.0	1
1.2	5.2	0
3.8	0.5	1
2.2	6.2	0
0.3	6.2	0

টেবিল 6.2.3

এখন আমরা $K = 3$ ধরে নিলে, আমরা সবচেয়ে কাছের তিনটি বিন্দু নিয়ে কাজ করব। ওপরের টেবিল থেকে আমরা প্রথম তিন সারির ডেটা পয়েন্ট নিলেই সবচেয়ে কাছের তিনটি বিন্দুর ক্লাস আমরা পেয়ে যাচ্ছি, যা যথাক্রমে হলো 0, 1, 0 অর্থাৎ দুটি বিড়াল ও একটি কুকুর (ধরে নিয়েছিলাম

$1 =$ কুকুর, $0 =$ বিড়াল)। যেহেতু বিড়ালের সংখ্যা বেশি, তাই আমাদের (2.2, 3) পয়েন্টটির ক্লাস হিসেবে আমরা 0 অর্থাৎ বিড়াল হিসেবে চিহ্নিত করব।

আবার যদি, $K = 5$ ধরে নিই, তাহলে আমরা সবচেয়ে কাছের পাঁচটি বিন্দু নিয়ে কাজ করব। ওপরের টেবিল থেকে আমরা প্রথম পাঁচ সারির ডেটা পয়েন্ট নিলেই সবচেয়ে কাছের ছয়টি বিন্দুর ক্লাস আমরা পেয়ে যাচ্ছি, যা যথাক্রমে হলো 0, 1, 0, 1, 1 অর্থাৎ দুটি বিড়াল ও তিনটি কুকুর। সুতরাং এই ক্ষেত্রে তাই আমাদের (2.2, 3) পয়েন্টটির ক্লাস হিসেবে আমরা 1 অর্থাৎ কুকুর অ্যাসাইন করব।

শেষ করার আগে, একটি বিষয় জেনে রাখা ভালো। যদি K -এর মান খুব ছোটো হয়, তখন ব্যাপারটি এরকম দাঁড়ায় যে কাছাকাছি অল্প দু-তিনটি ডেটা পয়েন্ট দেখেই আমরা সিদ্ধান্তে পৌছে যাই, এর ফলে বিভিন্ন ধরনের noise (বাজে ডেটা, যেগুলো আমাদের ডেটাসেটের কোনো অংশ নয়, তুলনাত্ত্ব) দিয়ে আমাদের সিদ্ধান্ত বায়সড (biased) বা পক্ষপাতদৃষ্ট হয়ে যেতে পারে। খুব সহজভাবে চিন্তা করুন, যদি আপনি মাত্র তিনজন মানুষকে জিজ্ঞাসা করেন যে, ‘চুরি করা কি ভালো?’ এবং দুর্ভাগ্যক্রমে ওই তিনজনের মধ্যে দুজন যদি হয় চোর এবং তারা আপনাকে ‘হ্যাঁ’ বলে এবং বাকি একজন ‘না’ বলে, তখন আপনি কী করবেন? যেহেতু দুজন ‘হ্যাঁ’ বলেছে, সেহেতু সেটিই আপনি মেনে নেবেন এবং হয়তো চুরি করা ভালো কাজ মনে করবেন, তাই না? তার মানে কী দাঁড়াল? এখানে চোর দুজনকে আমরা noise ডেটার সঙ্গে তুলনা করতে পারি। তাই, k -এর মান খুব ছোটো হলে noise ডেটা দিয়ে প্রভাবিত হওয়ার সম্ভাবনা বেশি থাকে। ফলে ভ্যারিয়েন্স বেড়ে যায়, বায়াস কমে যায়।

একই ভাবে, যদি k -এর মান আমরা অনেক বেশি নিই, ($\text{ধরণ } k = 1000$), তাহলে আবার সমস্যা হলো আগের ঘটনার উলটো ঘটনা ঘটবে, অর্থাৎ ভ্যারিয়েন্স কমে গিয়ে বায়াস বেড়ে যাবে। আর তাই, k -এর মান নির্ধারণ করার সময় ভারসাম্য বজায় রাখতে হয়, যাতে খুব বড়ও না হয়, আবার খুব ছোটোও না হয়।

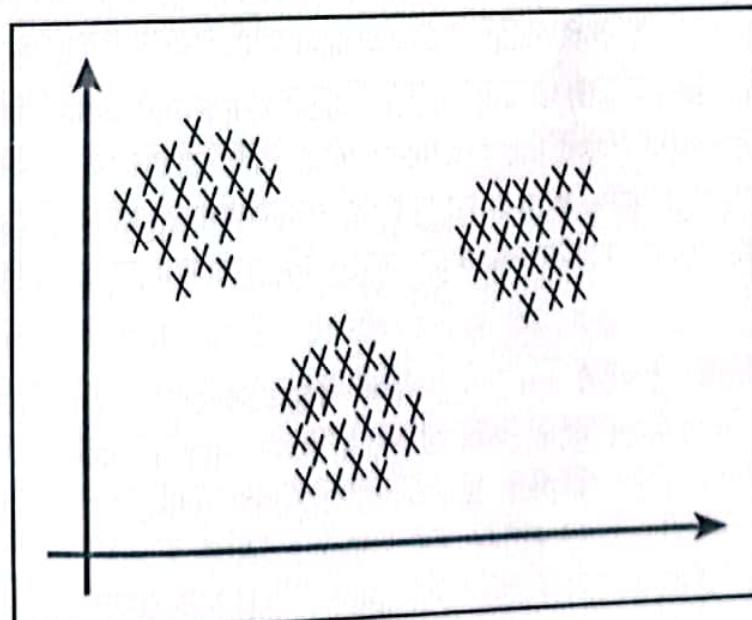
K -এর অপটিমাল (optimal) মান নেওয়ার একটি উপায় হচ্ছে, মোট যত ডেটা আছে, ($\text{ধরি } n$ -
সংখ্যক) তার বর্গমূলের মানকে আমরা K -এর অপটিমাল মান হিসেবে নিতে পারি। যদি, ধরা যাক,
 $n = 150$ হয়, তবে $K \cong \sqrt{150} \cong 12.28 \cong 13$ নিতে পারি (যেহেতু K -এর মান বেজোড় হবে,
তাই আমরা 12 নেব না, 13 নেব)।

এই ছিল KNN অ্যালগরিদমের বিবরণ। এটি আসলে খুবই ছোটো এবং সহজ একটি ক্লাসিফিকেশন অ্যালগরিদম। আশা করি, সবাই বুঝতে পেরেছেন কীভাবে KNN অ্যালগরিদমের সাহায্যে আমরা ক্লাসিফিকেশন করতে পারি।

অধ্যায় ৭ : কে-মিনস ক্লাস্টারিং (K-Means Clustering)

কে-মিনস ক্লাস্টারিং অ্যালগরিদমটি প্রথম ব্যবহার করেন ম্যাককুইন (James MacQueen) নামের একজন বিজ্ঞানী 1967 সালে, যদিও মূল ধারণাটি ছিল হ্যগো স্টেইনহাউস (Hugo Steinhaus, 1887 - 1972) নামে একজন বিজ্ঞানীর (1957 সাল)। ম্যাককুইন ছিলেন ক্যালিফোর্নিয়া বিশ্ববিদ্যালয়ের পরিসংখ্যান বিভাগের একজন অধ্যাপক।

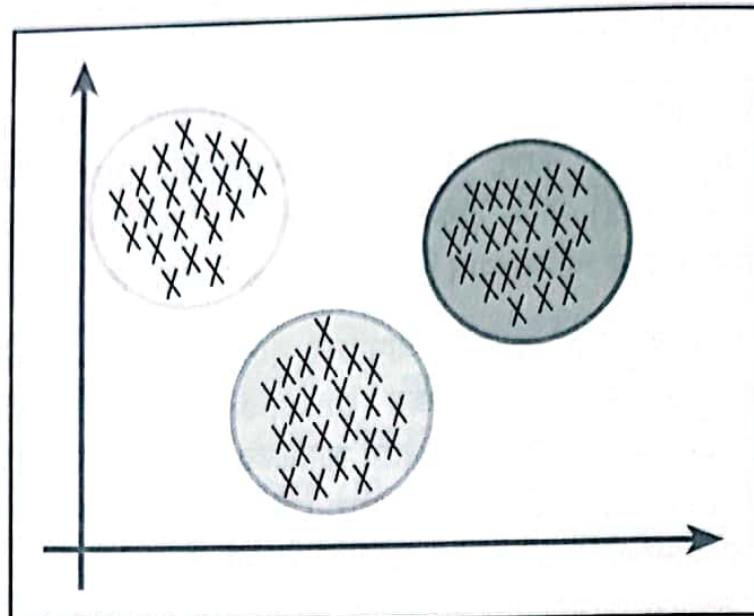
মূল অ্যালগরিদমে যাওয়ার আগে প্রথমেই আমাদের বুঝে নিতে হবে ক্লাস্টারিং আসলে কী? ক্লাস্টারিং হচ্ছে এক ধরনের আনসুপারভাইজড (Unsupervised) মেশিন লার্নিং পদ্ধতি, যেটি আমরা প্রথমেই আলোচনা করেছি। অর্থাৎ, এখানে আমাদের ট্রেনিং ডেটার কোনো লেবেল থাকবে না। আনলেবেলড (unlabeled) ট্রেনিং ডেটা দিয়ে মেশিনকে ট্রেনিং দেওয়া হবে এবং মেশিন চেষ্টা করবে সেইসব আনলেবেলড ডেটার ভেতরে কোনো ধরনের প্যাটার্ন খুঁজে বের করতে (unlabeled ডেটা সম্পর্কে আমরা প্রথমেই ধারণা দিয়েছিলাম পরিচ্ছেদ ২.২-এ)। তারপরে যে সব ডেটা পয়েন্ট একই প্যাটার্নের, তাদেরকে একটি গুচ্ছ/ক্লাস্টারে নিয়ে নেবে এই অ্যালগরিদম। এই হচ্ছে সাধারণভাবে বলতে গেলে ক্লাস্টারিং। নিচের ছবিটি দেখি :



ছবি 7.1

ঘবিতে, প্রতিটি ক্রস চিহ্ন হচ্ছে একটি ডেটা পয়েন্ট। সব পয়েন্টই যেহেতু এখানে সাদাকালো, তাহলে কিন্তু সবার চেহারা একই হয়ে গেল। এদের এখন আলাদা করা কিন্তু মুশকিল। এখন যদি বলা হয়, এই সাদাকালো পয়েন্টগুলোকে ডেটার বিন্যাস থেকে কোনো প্যাটার্ন বের করে তার

ওপরে ভিত্তি করে কতগুলো গুচ্ছ/ক্লাস্টারে ভাগ করতে, যাতে একই ধরনের প্যাটার্নের সব ডেটা একসঙ্গে থাকে, তখন আপনারা কী করতেন?



ছবি 7.2

আমার আন্দাজ বলছে, আপনারা অনেকটা এভাবে সবগুলো ডেটা তিনটি গুচ্ছ/ক্লাস্টারে ভাগ করে ফেলতেন, তাই না? ঠিকই আছে আপনাদের ভাগ করা। আমি হলেও এভাবেই করতাম।

এখন কথা হচ্ছে, এই ভাগ করাটা আমরা কীসের ভিত্তিতে করলাম? আমরা এই ভাগটা করলাম চোখের আন্দাজে। যে পয়েন্টগুলো একে অপরের কাছাকাছি, তাদেরকে আমরা একই ক্লাস্টারে নিয়ে নিয়েছি। এখন কথা হচ্ছে, আমরা না হয় চোখের আন্দাজে ছবির দিকে তাকিয়ে বুঝতে পারছি যে কোন পয়েন্ট কোন ক্লাস্টারের ভেতরে যাবে, কিন্তু সমস্যা হচ্ছে কম্পিউটারকে সেটি কী করে বোঝাই?

সেজন্যই আমাদের এখন কোনো একটি গাণিতিক পদ্ধতিতে যেতে হবে, যাতে করে আমাদের কম্পিউটার হিসাবনিকাশ করে বুঝে ফেলতে পারে কোন পয়েন্ট কোন ক্লাস্টারে যাবে। এই ক্লাস্টারিংয়ের জন্য আমরা খুব জনপ্রিয় একটি অ্যালগরিদম পড়ব, যার নাম হচ্ছে K-Means Clustering।

পরিচেদ ৭.১ : কে-মিনস ক্লাস্টারিংয়ের সংক্ষিপ্ত গাণিতিক বর্ণনা

বিস্তারিত বর্ণনায় যাওয়ার আগে, একটু সংক্ষেপে ব্যাখ্যা করে নিই যে কাজটি আসলে কীভাবে হবে। আমাদের যেটি করতে হবে, সেটি হচ্ছে প্রথমে আমাদেরকে যে ট্রেনিং ডেটা দেওয়া থাকবে,

অধ্যায় ৭ : কে-মিনস ক্লাস্টারিং (K-Means Clustering)

খেলান থেকে ঠিক করে নিতে হবে যে আমরা কয়টি ক্লাস্টার নিয়ে কাজ করতে চাই। ধরা যাক, আমরা K-সংখ্যক ক্লাস্টার নিয়ে কাজ করতে চাই। প্রতিটি ক্লাস্টারের একটি কেন্দ্রবিন্দু বা সেন্টার পয়েন্ট থাকবে, যেটাকে আমরা বলব সেন্ট্রয়েড (centroid)। এই সেন্ট্রয়েডের মান হবে ওই ক্লাস্টারে যতগুলো পয়েন্ট আছে সবগুলোর গড় মান।

আমরা শুরুতে প্রতিটি সেন্ট্রয়েডকে র্যানডমলি একটি করে মান দিয়ে দেব। এরপর, আমাদের ট্রেনিং ডেটা থেকে আমরা একটি করে ডেটা পয়েন্ট নেব এবং K-সংখ্যক সেন্ট্রয়েডের প্রত্যেকটি থেকে সেই ডেটা পয়েন্টের দূরত্ব মাপব। যেই সেন্ট্রয়েড সবচেয়ে কাছে থাকবে, ওই ডেটা পয়েন্টকে সেই সেন্ট্রয়েডের জন্য যে ক্লাস্টার আছে তাতে অ্যাসাইন করব। এভাবে আমরা একটি করে ডেটা পয়েন্ট নেব এবং তাকে কোনো-না-কোনো ক্লাস্টারে অ্যাসাইন করব।

সেই সঙ্গে আরো একটি কাজ চলবে, সেটি হচ্ছে, প্রতিবার ক্লাস্টারে একটি করে পয়েন্ট যুক্ত হওয়ার পর, আবার নতুন করে ওই ক্লাস্টারে থাকে সবগুলো পয়েন্টের গড় নিতে হবে। এই নতুন গড়-ই হবে তখন ওই ক্লাস্টারের জন্য নতুন সেন্ট্রয়েড, যেটি আমাদের প্রতিবার হিসাব করতে হবে।

অর্থাৎ, আমাদের কাজ হবে দুটি – ক্লাস্টার অ্যাসাইনমেন্ট এবং সেন্ট্রয়েড আপডেট।

ধরা যাক, আমরা m-সংখ্যক ট্রেনিং ডেটার প্রতিটি পয়েন্ট একটি করে ক্লাস্টারে অ্যাসাইন করব। ধরা যাক, আমরা m-সংখ্যক ক্লাস্টারের আছে। আমরা একটি অ্যারের কথা চিন্তা করি, যার নাম C এবং আমাদের K-সংখ্যক ক্লাস্টারের আছে। আমরা একটি অ্যারের কথা চিন্তা করি, যার নাম C এবং আমাদের m-সংখ্যক উপাদান থাকবে। অ্যারের i-তম উপাদান হবে i-তম সাইজ হচ্ছে m। এই অ্যারেতে m-সংখ্যক উপাদান থাকবে। অ্যারের i-তম উপাদান হবে C⁽ⁱ⁾। ট্রেনিং ডেটাকে 1 থেকে K-এর মধ্যে কত নম্বর ক্লাস্টারে অ্যাসাইন করা হয়েছে সেই সংখ্যাটি।

ধরা যাক, আমাদের প্রথম ডেটা পয়েন্ট হলো a, যাকে 2 নম্বর ক্লাস্টারে অ্যাসাইন করা হয়েছে। একইভাবে মুক্তরাঃ, C⁽¹⁾ হবে 2 যেখানে C⁽¹⁾ দিয়ে প্রথম ডেটা পয়েন্টকে নির্দেশ করা হচ্ছে। একইভাবে যদি দ্বিতীয় ডেটা পয়েন্টকে 4 নম্বর ক্লাস্টারে অ্যাসাইন করা হয়, তাহলে C⁽²⁾ হবে 4। কোনো ডেটা পয়েন্টকে K-সংখ্যক ক্লাস্টারের মধ্যে সেই ক্লাস্টারকেই অ্যাসাইন করা হবে, যে ক্লাস্টারের সেন্ট্রয়েড (ওই ক্লাস্টারে অ্যাসাইন করা সব পয়েন্টের মানের গড়মান) থেকে ওই ডেটা পয়েন্টের দূরত্ব অন্য সব ক্লাস্টারের সেন্ট্রয়েডের তুলনায় সবচেয়ে কম হবে। আমরা যদি k-তম ক্লাস্টারের সেন্ট্রয়েডকে μ_k দিয়ে চিহ্নিত করি, তাহলে i-তম ট্রেনিং ডেটা $x^{(i)}$ এবং k-তম ক্লাস্টারের সেন্ট্রয়েড μ_k -এর মধ্যে দূরত্ব হবে $\|x^{(i)} - \mu_k\|$ যেখানে $1 \leq i \leq m$ ।

এই গেল আমাদের প্রথম ধাপ – ক্লাস্টার অ্যাসাইনমেন্ট। এরই সঙ্গে আরেকটি ধাপ আছে, সেটি হচ্ছে সেন্ট্রয়েডের মান আপডেট করা। প্রতিবার একটি করে নতুন ডেটা পয়েন্ট $x^{(i)}$ কোনো একটি ক্লাস্টার k-তে অ্যাসাইন করার পর, সেই ক্লাস্টারের সেন্ট্রয়েড আপডেট করতে হবে। নতুন

সেন্ট্রয়েডের মান হবে ওই ক্লাস্টারে আয়াসাইন করা সব ট্রেনিং ডেটার মানের (সেদ্য আয়াসাইন করা নতুন ডেটা পয়েন্টসহ) গড়।

আমরা যদি এখন এর জন্য কস্ট ফাংশন লিখতে যাই, সেটি তাহলে হবে এরকম –

$$J(\underbrace{C^{(1)}, C^{(2)}, C^{(3)}, \dots C^{(m)}}_{m \text{ number of training examples}}, \underbrace{\mu_1, \mu_2, \mu_3, \dots \mu_K}_{k \text{ number of cluster centroids}}) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_k\|^2$$

এবং অবজেক্টিভ ফাংশন হবে,

$$\underset{(c^{(i)}, \mu)}{\text{minimize}} J(C^{(1)}, C^{(2)}, C^{(3)}, \dots C^{(m)}, \mu_1, \mu_2, \mu_3, \dots \mu_K)$$

পরবর্তী পরিচ্ছেদে উদাহরণ দিয়ে কে-মিনস ক্লাস্টারিং আরে বিস্তারিত আলোচনা করা হয়েছে।
উদাহরণটি দেখলেই অনেকখানিই পরিষ্কার ধারণা পাওয়া যাবে।

পরিচ্ছেদ ৭.২ : উদাহরণ

যাক, অনেক গণিত হলো, এখন চলুন একটি ছোটো উদাহরণ দিয়ে দেখি কীভাবে এই কে-মিনস ক্লাস্টারিং কাজ করে।

X	Y
2	4
2	3
5	2
6	2
5	2.5
2.5	3.5

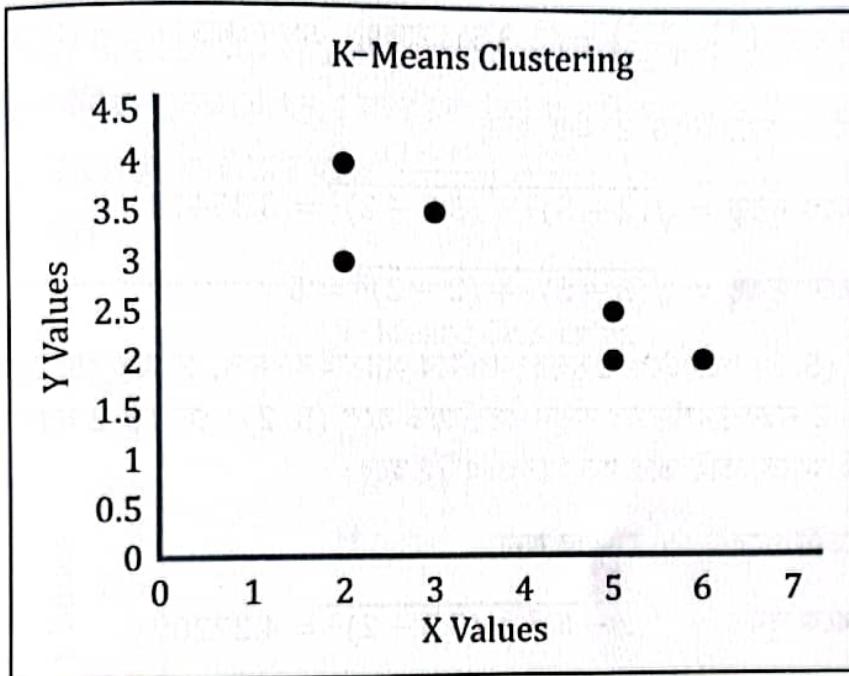
টেবিল 7.2.1

টেবিল 7.2.1-এর ডেটা একটি গ্রাফে প্লট করলে অনেকটা গ্রাফ 7.2.1-এর মতো দেখাবে। এখন দেখুন, আমরা যদি এই ডেটা সেটের ওপরে কে-মিনস ক্লাস্টারিং অ্যালগরিদম প্রয়োগ করতে যাই, তাহলে প্রথমেই আমাদের ঠিক করতে হবে আমরা কয়টি ক্লাস্টার নেব, অর্থাৎ K-এর মান কত হবে? আমরা যে-কোনো মান দিয়েই শুরু করতে পারি। ধরে নিই K = 2 (চিত্র দেখেই বোঝা

অধ্যায় ৭ : কে-মিনস ক্লাস্টারিং (K-Means Clustering)

যাচ্ছে ক্লাস্টার ২টি হবে, তাই দুটি ক্লাস্টার দিয়েই দেখাচ্ছি। তবে K -এর মান কত হলে সবচেয়ে ভালো হবে সেটি বের করার একটি উপায় আছে, সেটি পরে আলোচনা করব।

যদি $K = 2$ নিই, তাহলে আমাদের দুটি ক্লাস্টারের জন্য দুটি সেন্ট্রয়েড হবে μ_1 ও μ_2 । এদের মান হিসেবে শুরুতে আমরা যে-কোনো আনুমানিক মান ধরে নিতে পারি। তবে ভালো বৃদ্ধি হচ্ছে আমাদের ট্রেনিং ডেটা পয়েন্টগুলো মধ্যে থেকে র্যানডমলি যে-কোনো দুটি পয়েন্টকে সেন্ট্রয়েড হিসেবে ধরে নেওয়া।



গ্রাফ 7.2.1

তাই, হিসাবের সুবিধার্থে আমরা ধরে নিই যে আমাদের $\mu_1 = (2, 4)$ এবং $\mu_2 = (5, 2)$ । এখন, প্রতিটি ডেটা পয়েন্ট থেকে আমরা এই দুটি সেন্ট্রয়েডের দূরত্ব বের করব এবং যেই সেন্ট্রয়েড থেকে দূরত্ব কম, আমরা ওই পয়েন্টকে সেই সেন্ট্রয়েডের ক্লাস্টারে অ্যাসাইন করব এবং সেই সেন্ট্রয়েডের মান আপডেট করব।

এখন, প্রথম ট্রেনিং ডেটা পয়েন্ট $(2, 4)$ -এর জন্য,

$$\mu_1 \text{ থেকে দূরত্ব} = 0$$

$$\mu_2 \text{ থেকে দূরত্ব} = \sqrt{(5 - 2)^2 + (2 - 4)^2} = 3.60555$$

গুরুত্বাং আমরা $(2, 4)$ পয়েন্টকে ১ নম্বর ক্লাস্টারে অ্যাসাইন করব এবং নতুন সেন্ট্রয়েড হবে $(2, 4)$ । যেহেতু ১ নম্বর ক্লাস্টারে মাত্র একটি পয়েন্টই আছে, তাই তার গড় মানও এটিই হবে।

এখন, দ্বিতীয় ট্রেনিং ডেটা পয়েন্ট $(2, 3)$ -এর জন্য,

$$\mu_1 \text{ থেকে দূরত্ব} = \sqrt{(2-2)^2 + (4-3)^2} = 1$$

$$\mu_2 \text{ থেকে দূরত্ব} = \sqrt{(5-2)^2 + (2-3)^2} = 3.16$$

সুতরাং আমরা $(2, 3)$ পয়েন্টকেও 1 নম্বর ক্লাস্টারে অ্যাসাইন করব, যেহেতু $(2, 3)$ থেকে μ_1 -এর দূরত্ব সর্বনিম্ন। সুতরাং, 1 নম্বর ক্লাস্টারে পয়েন্ট আছে এখন দুটি : $(2, 4)$ ও $(2, 3)$ । সুতরাং, নতুন সেন্ট্রয়েড হবে, $\left(\frac{2+2}{2}, \frac{4+3}{2}\right) = (2, 3.5)$ । সুতরাং, এখন থেকে $\mu_1 = (2, 3.5)$ ।

এরপর তৃতীয় ডেটা পয়েন্ট $(5, 2)$ এর জন্য,

$$\mu_1 \text{ থেকে দূরত্ব} = \sqrt{(2-5)^2 + (3.5-2)^2} = 3.35410$$

$$\mu_2 \text{ থেকে দূরত্ব} = \sqrt{(5-5)^2 + (2-2)^2} = 0$$

সুতরাং আমরা $(5, 2)$ পয়েন্টকে 2 নম্বর ক্লাস্টারে অ্যাসাইন করব, যেহেতু $(5, 2)$ থেকে μ_2 -এর দূরত্ব সর্বনিম্ন। 2 নম্বর ক্লাস্টারের নতুন সেন্ট্রয়েড হবে $(5, 2)$ । যেহেতু 2 নম্বর ক্লাস্টারে মাত্র একটি পয়েন্টই আছে, তাই তার গড় মানও এটিই হবে।

এরপর চতুর্থ ডেটা পয়েন্ট $(6, 2)$ -এর জন্য,

$$\mu_1 \text{ থেকে দূরত্ব} = \sqrt{(2-6)^2 + (3.5-2)^2} = 4.27200$$

$$\mu_2 \text{ থেকে দূরত্ব} = \sqrt{(5-6)^2 + (2-2)^2} = 1$$

সুতরাং আমরা $(6, 2)$ পয়েন্টকেও 2 নম্বর ক্লাস্টারে অ্যাসাইন করব, যেহেতু $(6, 2)$ থেকে μ_2 -এর দূরত্ব সর্বনিম্ন। সুতরাং, 2 নম্বর ক্লাস্টারে পয়েন্ট আছে এখন দুটি : $(5, 2)$ ও $(6, 2)$ । সুতরাং, নতুন সেন্ট্রয়েড হবে, $\left(\frac{5+6}{2}, \frac{2+2}{2}\right) = (5.5, 2)$ । সুতরাং, এখন থেকে $\mu_2 = (5.5, 2)$ ।

এরপর পঞ্চম ডেটা পয়েন্ট $(5, 2.5)$ -এর জন্য,

$$\mu_1 \text{ থেকে দূরত্ব} = \sqrt{(2-5)^2 + (3.5-2.5)^2} = 3.16228$$

$$\mu_2 \text{ থেকে দূরত্ব} = \sqrt{(5.5-5)^2 + (2-2.5)^2} = 0.70710$$

সুতরাং আমরা $(5, 2.5)$ পয়েন্টকেও 2 নম্বর ক্লাস্টারে অ্যাসাইন করব, যেহেতু $(5, 2.5)$ থেকে μ_2 -এর দূরত্ব সর্বনিম্ন। সুতরাং, 2 নম্বর ক্লাস্টারে পয়েন্ট আছে এখন তিনটি : $(5, 2)$, $(6, 2)$ ও

অধ্যায় ৭ : কে-মিনস ক্লাস্টারিং (K-Means Clustering)

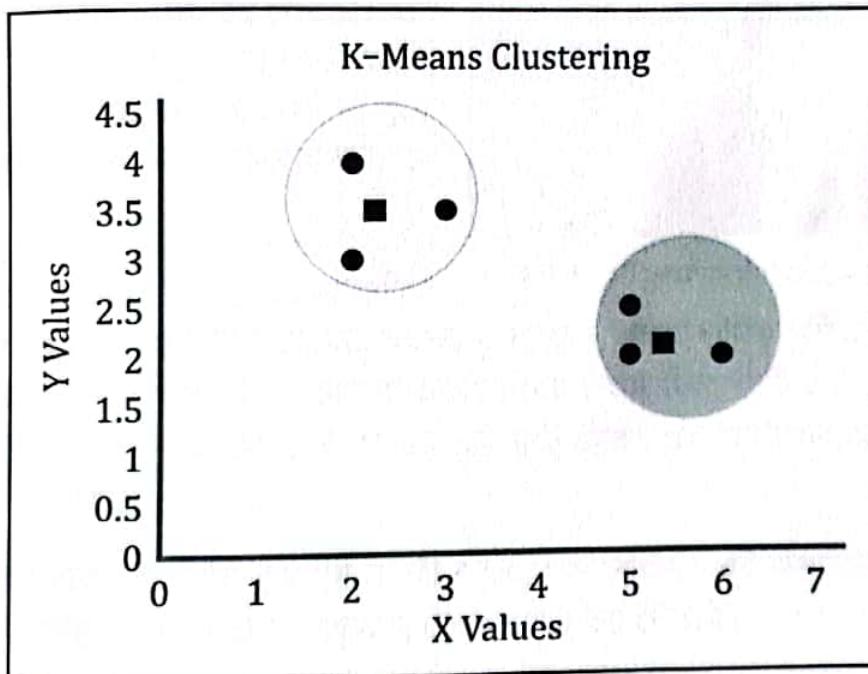
$(5, 2.5)$ । সুতরাং নতুন সেন্ট্রয়েড হবে, $\left(\frac{5+6+5}{3}, \frac{2+2+2.5}{3}\right) = (5.33, 2.17)$ । সুতরাং এখন $\mu_2 = (5.33, 2.17)$ ।

সরশেয়ে, ষষ্ঠ ডেটা পয়েন্ট $(2.5, 3.5)$ -এর জন্য,

$$\mu_1 \text{ থেকে দূরত্ব} = \sqrt{(2 - 2.5)^2 + (3.5 - 3.5)^2} = 0.5$$

$$\mu_2 \text{ থেকে দূরত্ব} = \sqrt{(5.33 - 2.5)^2 + (2.17 - 3.5)^2} = 2.49799$$

সুতরাং $(2.5, 3.5)$ পয়েন্টকে আমরা 1 নম্বর ক্লাস্টারে অ্যাসাইন করব, যেহেতু μ_1 থেকে $(2.5, 3.5)$ -এর দূরত্ব সর্বনিম্ন। সুতরাং এখন, 1 নম্বর ক্লাস্টারে পয়েন্ট আছে তিনটি : $(2, 4), (2, 3)$ ও $(2.5, 3.5)$ । সুতরাং নতুন সেন্ট্রয়েড হবে, $\left(\frac{2+2+2.5}{3}, \frac{4+3+3.5}{3}\right) = (2.17, 3.5)$ । সুতরাং এখন $\mu_1 = (2.17, 3.5)$ ।



গ্রাফ 7.2.2

এখন যদি ওপরের গ্রাফটি দেখি, তাহলে যে বর্গাকৃতির চিহ্ন দুটি দেখব, সে দুটি হচ্ছে আমাদের দুটি ক্লাস্টারের নতুন সেন্ট্রয়েড। বাঁ দিকের ক্লাস্টারটি হচ্ছে 1 নম্বর ক্লাস্টার এবং ডান দিকের ক্লাস্টারটি হচ্ছে 2 নম্বর ক্লাস্টার।

আগে বলেছিলাম যে, K-এর মান কীভাবে নির্ণয় করতে হয় সে সম্পর্কে বলব। ব্যাপারটি হচ্ছে, K-এর মান হিসেবে কোনটি ব্যবহার করলে কষ্ট সবচেয়ে কম হবে, সেটি সরাসরি বের করার জন্য কোনো গাণিতিক সূত্র নেই। তাই যেটি করতে হবে, K-এর মান 2, 3, 4, 5... ইত্যাদি ধরে দেখতে

মেশিন লার্নিং অ্যালগরিদম

হবে যে K-এর কোন মানের জন্য কস্ট ফাংশন কনভার্জ করছে, অর্থাৎ সর্বনিম্ন মান দিচ্ছে। এটি একটি ট্রায়াল অ্যান্ড এরর (Trial and error) পদ্ধতি হবে, এই আর কি। খুব ভালো হয়, যদি আপনারা K-বনাম-কস্ট-এর একটি গ্রাফ তৈরি করে নিতে পারেন, তাহলে আপনাদের ব্যাপারটি উপলব্ধি করতে সুবিধা হবে।

আশা করি, সবাই বুঝতে পেরেছেন কীভাবে কে-মিনস ক্লাস্টারিং অ্যালগরিদমের সাহায্যে আমরা ক্লাস্টারিং করতে পারি।

অধ্যায় ৮ : নাইভ বেইজ ক্লাসিফায়ার (Naive Bayes Classifier)

যশা করি, আপনারা সবাই উচ্চমাধ্যমিক গণিতে বেইজ থিওরেম পড়েছেন। আমার যতদূর মনে পড়ে, উচ্চমাধ্যমিক গণিত বইয়ে নষ্ট বল্টু তৈরি করা মেশিন ইত্যাদি সম্পর্কিত কোনো একটি বিখ্যাত অঙ্ক ছিল, যেটি এই বেইজ থিওরেম দিয়ে করতে হতো। ওই অঙ্ক দেখলেই আমার ঘাম ছুটে যেত। ওই অধ্যায়ের অঙ্কগুলো করতে আমি ভীষণ ভয় পেতাম। আর তার কারণ একটিই, আমি বুঝতাম না কী করছি, কেন করছি। এই অধ্যায়ের অঙ্কগুলো না বুঝে অনেকটা মুখস্থ করার মতো করেছিলাম, সে কারণেই ভয় পেতাম।



ছবি 8.1 : Thomas Bayes (1702-1761)

যশা করি, আপনারা সবাই উচ্চমাধ্যমিক গণিতে বেইজ থিওরেম পড়েছেন। আমার যতদূর মনে পড়ে, উচ্চমাধ্যমিক গণিত বইয়ে নষ্ট বল্টু তৈরি করা মেশিন ইত্যাদি সম্পর্কিত কোনো একটি বিখ্যাত অঙ্ক ছিল, যেটি এই বেইজ থিওরেম দিয়ে করতে হতো। ওই অঙ্ক দেখলেই আমার ঘাম ছুটে যেত। ওই অধ্যায়ের অঙ্কগুলো করতে আমি ভীষণ ভয় পেতাম।

আর তার কারণ একটিই, আমি বুঝতাম না কী করছি, কেন করছি। এই অধ্যায়ের অঙ্কগুলো না বুঝে অনেকটা মুখস্থ করার মতো করেছিলাম, সে কারণেই ভয় পেতাম।

স্ন্তান্বযতা (Probability) এখন আমার অনেক প্রিয় একটি বিষয়। এটি আমার পড়তেও ভালো লাগে, পড়তেও ভালো লাগে। আর এই নাইভ বেইজ ক্লাসিফায়ার বুঝতে হলে আমাদেরকে উচ্চ মাধ্যমিকে পড়ে আসা বেইজ থিওরেম এবং সেই সঙ্গে স্ন্তান্বযতার কিছু জিনিস একটু ঝালাই করে নিতে হবে।

কিন্তু তার আগে একটু বলে নিই, বেইজ থিওরেমের জনক রেভারেন্ড থমাস বেইজ (Thomas Bayes, 1702-1761)। তাঁর নামেই এর নামকরণ। থমাসের কিছু অসমাঞ্ছ গবেষণার হাত ধরেই উচ্ছিবিত হয় এই থিওরেমের।

পরিচ্ছদ ৮.১ : সন্তাব্যতার টুকিটাকি

খুব সহজ করে বলতে গেলে, সন্তাব্যতা হচ্ছে কোনো ঘটনা ঘটার সন্তাবনার একটি গাণিতিক প্রকাশ। যেখানেই আপনারা সন্তাব্যতা পাবেন, সেখানেই দেখবেন তার সঙ্গে কোনো সংখ্যা ও গাণিতিক প্রকাশ জড়িত। যদি বলি, আজকে বৃষ্টি হওয়ার সন্তাব্যতা 70% , সেটি গাণিতিক সন্তাব্যতার একটি উদাহরণ। এখন, এই সন্তাব্যতার মান 0 থেকে 1 -এর ভেতরে যে-কোনো সংখ্যা হতে পারে। 0 মানে ঘটনাটি ঘটার কোনো সন্তাবনাই নেই, আর 1 মানে ঘটনাটি ঘটবেই, কোনো নড়চড় হবে না। সন্তাব্যতা 0.5 মানে অর্ধার্ধি সন্তাবনা আছে ঘটনাটি ঘটার। যেমন, আপনি যদি একটি কয়েন ছুড়ে মারেন ওপরে টস করার জন্য, 50% সন্তাবনা আছে Head আসার, আর 50% সন্তাবনা আছে Tail আসার। অর্থাৎ Head আসার Probability 0.5 ।

এই গেল, সন্তাব্যতার মান কত থেকে কত হতে পারে, তার একটি বর্ণনা। এখন আসি কীভাবে সন্তাব্যতা নির্ণয় করবেন সে উপায়ে। সন্তাব্যতা বের করার সূত্র হচ্ছে =

$$\frac{\text{Number of Events occurred in Favor of Expected Outcome}}{\text{Number of total events occurred irrespective of Expected Outcome}}$$

এ কথার মানে কী? একটি লুভুর ছক্কার কথা চিন্তা করুন। একটি ছক্কা যদি আমরা চালি, তাহলে আমরা কয় ধরনের ভিন্ন ভিন্ন মান পেতে পারি? 6 ধরনের, তাই না? $1, 2, 3, 4, 5$ ও 6 । তার মানে, একটি ছক্কা চাললে মোট ছয় ধরনের ঘটনা ঘটতে পারে।

এখন যদি আমরা চাই যে আমাদের ছক্কায় 4 উঠুক, সেটি কয়টি ঘটনার জন্য ঘটতে পারে? ছক্কার ওপরে $1, 2, 3$ ইত্যাদি থাকলে কি আমরা 4 উঠেছে বলে ধরে নেব? মোটেই না। ছক্কায় ওপরে শুধু 4 উঠলেই কেবল আমরা আমাদের প্রত্যাশিত ফলাফল (Expected Outcome) পেতে পারি। সুতরাং, সূত্র অনুযায়ী একটি ছক্কা চাললে তাতে 4 ওঠার সন্তাবনা $\frac{1}{6}$ । আবার আমরা যদি চাই 6 আমাদের ছক্কায় শুধু বেজোড় মান উঠুক, অর্থাৎ $1, 3$ কিংবা 5 উঠুক, তাহলে তার সন্তাব্যতা হবে $\frac{3}{6} = \frac{1}{2}$ ।

এই গেল আমাদের সন্তাব্যতা কীভাবে নির্ণয় করতে হয় তার পদ্ধতি। আরেকটি বিষয় আমাদের জেনে নিতে হবে, সেটি হচ্ছে, শর্তাধীন সন্তাব্যতা বা কন্ডিশনাল প্রোবাবিলিটি (Conditional Probability)। অর্থাৎ, আপনাকে কোনো একটি ঘটনা (বা, শর্ত) দিয়ে দেওয়া হবে, সেটি ঘটেছে ধরে নিয়ে সেই ঘটনার সাপেক্ষে আপনাকে অন্য কোনো ঘটনা ঘটার সন্তাব্যতা বিচার করতে হবে। লুভুর ছক্কা দিয়েই বোঝাই। ধরা যাক, আপনাকে একটি সাধারণ ছক্কা দিয়ে বলল তাতে 5 ওঠার সন্তাব্যতা কত? আপনি সঙ্গে সঙ্গে উত্তর দিয়ে দিতে পারবেন, $\frac{1}{6}$, তাই না? কিন্তু যদি এখন আপনাকে বলে দেওয়া হয়, আপনি ছক্কা চাললে আপনার কোনো জোড় সংখ্যা উঠবে না। এবার

অধ্যায় ৮ : নাইভেজ ক্লাসিফায়ার (Naive Bayes Classifier)

যদি বলা হয় এই ঘটনার সাপেক্ষে ছক্কা চেলে 5 পাওয়ার সম্ভাবনা কত তা বের করতে, তখন ইত্তাবে সেটি হিসাব করবেন?

দেখুন, আপনাকে যেহেতু বলেই দেওয়া হচ্ছে যে আপনার কোনো জোড় সংখ্যা উঠবে না, তার মানে দাঁড়াচ্ছে ছক্কা চেলে 2, 4 ও 6 পাওয়ার কোনো সম্ভাবনা-ই নেই। তাহলে আপনার ছক্কা চেলে উঠতে পারে 1, 3 কিংবা 5। তাহলে এই ক্ষেত্রে আপনার 5 পাওয়ার সম্ভাব্যতা বেড়ে দাঁড়াবে $\frac{1}{3}$ । একইভাবে যদি আপনাকে বলে দেওয়া হতো যে, আপনি ছক্কা চাললে আপনার 4 বাদে অন্য যে-কোনো সংখ্যা উঠতে পারে, তাহলে সে ক্ষেত্রে আপনার 5 পাওয়ার সম্ভাবনা হতো $\frac{1}{5}$ । এটিই হচ্ছে শর্তাধীন সম্ভাব্যতা।

যদি x কোনো ঘটনা বোঝায়, যার মানে হচ্ছে, 'c is Not Even' এবং c দিয়ে ছক্কার ওপরে কোন মান উঠবে সেটি নির্দেশ করা হয়, তাহলে আমাদের ছক্কায় জোড় সংখ্যা উঠবে না, এই শর্তসাপেক্ষে হচ্ছে 5 পড়ার সম্ভাব্যতাকে লেখা হবে $P(c = 5 | x)$; যেখানে x হচ্ছে আমাদের আগে থেকে দিয়ে দেওয়া শর্ত এবং $P(c = 5 | x)$ মানে বোঝাচ্ছে এই শর্তের অধীন থাকা অবস্থায় $c = 5$ হওয়া সম্ভাব্যতা। সুতরাং,

$$P(c = 5 | x) = \frac{1}{3}$$

পরিচ্ছেদ ৮.২ : বেইজ থিওরেম হাতে-কলমে

এখন আসি বেইজ থিওরেমে। সাধারণত, শর্তাধীন সম্ভাব্যতার ক্ষেত্রে আমরা আগে কোনো একটি ঘটনা ঘটে গেছে, তার সাপেক্ষে পরের ঘটনা ঘটার সম্ভাব্যতা কত সেটি বের করি।

যদি, বৃষ্টি হয়েছে, তার সাপেক্ষে রাস্তায় কাদা-পানি থাকার সম্ভাবনা কত? কিংবা, ক্যানসার হয়েছে, তার সাপেক্ষে রোগীর মৃত্যুর সম্ভাব্যতা কত? ইত্যাদি।

বিষ্টু, বেইজ থিওরেমে এর বিপরীত সম্ভাব্যতা বের করা হয়। পরের যে ঘটনাটি ঘটেছে, তার সম্ভাব্যতা আমরা জানি, তার সাপেক্ষে আগের কোনো একটি ঘটনা ঘটার সম্ভাব্যতা কত (অর্থাৎ, পরের ঘটনাটি আগের কোনো একটি নির্দিষ্ট ঘটনার কারণে ঘটেছে তার সম্ভাব্যতা কত), সেটি আমাদের বের করতে হয়। অর্থাৎ, রাস্তায় কাদা-পানি দেখা যাচ্ছে, এখন এই কাদা-পানি যে বৃষ্টির কারণেই হয়েছে, রাস্তায় কেউ এক বালতি পানি ছুড়ে মারার কারণে হয়নি, তার সম্ভাব্যতা কত; কিংবা, রোগী মারা গিয়েছে, কিন্তু রোগী যে ক্যানসারজনিত কারণেই মারা গিয়েছে, অন্য কোনো রোগের কারণে নয়, তার সম্ভাব্যতা কত, এরকম একটি বিষয়।

মেশিন লার্নিং অ্যালগরিদম

আমরা আগের ঘটনাকে যদি x ধরি এবং পরের ঘটনাকে যদি c ধরি এবং বেইজ থিওরেম প্রয়োগ করে যদি পরের ঘটনাটি যে আগের ঘটনার কারণেই ঘটেছে তার সন্তান্যতা বের করতে চাই, তাহলে সূত্র হবে এরকম –

$$P(X | C) = \frac{P(C | X) \cdot P(X)}{P(C)}$$

খুব ছোট একটি উদাহরণ দিয়ে বেইজ থিওরেম বোঝা শেষ করি। ধরা যাক, কোনো এলাকায় এক সমীক্ষা থেকে জানা যায়, সেখানকার এলাকাবাসীর ক্যানসার হওয়ার ঝুঁকি রয়েছে এবং প্রত্যেকের ক্যানসার হওয়ার সন্তান্যতা শতকরা 60 ভাগ। এ ছাড়াও, আরো জানা যায় যে ক্যানসারজনিত কারণে আগামী দুই বছরের মধ্যে ওই এলাকার কারো মারা যাওয়ার সন্তান্যনা শতকরা 85 ভাগ এবং ক্যানসার না হয়ে মারা যাওয়ার সন্তান্যনা শতকরা 45 ভাগ।

এখন, ধরা যাক, ওই এলাকায় ‘ C ’ নামে এক বয়স্ক ভদ্রলোক থাকতেন, যিনি ওই সমীক্ষা চালানোর ঠিক এক বছরের মাথায় মারা যান। এখন আপনাকে বের করতে হবে যে ওই ভদ্রলোক ক্যানসারজনিত কারণে মারা গিয়েছেন, তার সন্তান্যতা কত?

এটি কীভাবে বের করবেন? প্রথমে বের করতে হবে যে আগের এবং পরের ঘটনা কোনটি? এখানে ঘটনা আছে দুটি – ক্যানসার হওয়া এবং মারা যাওয়া। কোনটি আগে হবে বলুন তো? ক্যানসার হয়ে মানুষ মারা যাবে, নাকি মানুষ মারা যাওয়ার পরে তাঁর ক্যানসার হবে? অবশ্যই প্রথমটি, তাই না?

তার মানে, এ ক্ষেত্রে,

আগের ঘটনা, x = ক্যানসার হওয়া, এবং

পরের ঘটনা, c = মারা যাওয়া।

আমাদেরকে বের করতে হবে $P(X | C)$ কত?

প্রথমে চলুন বের করি, $P(C | X)$ কত?

$P(C | X)$ হচ্ছে ক্যানসার হওয়ার পরে মারা যাওয়ার সন্তান্যতা $= 85\% = 0.85$ ।

এর পরে, $P(X) = ক্যানসার হওয়ার সন্তান্যতা = 60\% = 0.60$ ।

আর সবশেষে $P(C) = মারা যাওয়ার সন্তান্যতা$

$= (\text{ক্যানসার হওয়ার সন্তান্যতা} \times \text{ক্যানসারের কারণে মৃত্যুর সন্তান্যতা})$

$+ (\text{ক্যানসার না হওয়ার সন্তান্যতা} \times \text{ক্যানসার না হয়েই মৃত্যুর সন্তান্যতা})$

অধ্যায় ৮ : নাইভ বেইজ ক্লাসিফায়ার (Naive Bayes Classifier)

$$\begin{aligned}
 &= (0.6 \times 0.85) + ((1 - 0.6) \times 0.45) \\
 &= 0.51 + 0.18 \\
 &= 0.69
 \end{aligned}$$

সুতরাং 'ক'-এর মৃত্যু যে ক্যানসারেই হয়েছে, তার সম্ভাব্যতা,

$$P(X | C) = \frac{0.85 \times 0.6}{0.69} = 0.739 = 73.9\%$$

যাক, আশা করি আপনারা বেইজ থিওরেম কীভাবে কাজ করে এবং কীভাবে প্রয়োগ করতে হয় সবাই কমবেশি বুঝতে পেরেছেন। এখন আমরা দেখব কীভাবে বেইজ থিওরেম ব্যবহার করে নাইভ বেইজ ক্লাসিফায়ার তৈরি ও ডেটাসেটের ওপরে ব্যবহার করতে হয়।

পরিচেদ ৮.৩ : হাতে কলমে নাইভ বেইজ ক্লাসিফায়ার

আমরা প্রথমে একটি ছোটো ডেটাসেট নিই :

Day	Outlook	Temperature	Routine	Wear Coat?
D_1	Sunny	Cold	Indoor	No
D_2	Sunny	Warm	Outdoor	No
D_3	Cloudy	Warm	Indoor	No
D_4	Sunny	Warm	Indoor	No
D_5	Cloudy	Cold	Indoor	Yes
D_6	Cloudy	Cold	Outdoor	Yes
D_7	Sunny	Cold	Outdoor	Yes

টেবিল 8.3.1

ডেটাসেটটি যদি আপনারা ভালো করে লক্ষ করেন, তাহলে দেখবেন যে মধ্যের তিনটি কলাম হচ্ছে আমাদের ফিচার ডেটা, আর শেষের কলামটি হচ্ছে আমাদের সর্বশেষ যে ফলাফল হবে সেটি।

এখন, আমাদের বের করতে হবে যে, যদি আমাদের ফিচার সেট এরকম হয় – {Cloudy, Warm, Outdoor} তাহলে ফলাফল কী হবে? বাইরে বের হওয়ার সময় কোট পরব, নাকি পরব না?

এটি বের করার জন্য আমরা যদি নাইড বেইজ ক্লাসিফায়ার প্রয়োগ করতে চাই, তাহলে
আমাদেরকে দুটি মান বের করতে হবে –

1. $P(C = Yes | X = \text{cloudy, warm, outdoor})$ এবং
2. $P(C = No | X = \text{cloudy, warm, outdoor})$ ।

এখানে $C = yes / no$ দিয়ে *Wear Coat*-এর মান *yes / no* বোঝানো হয়েছে।

প্রথমটির ক্ষেত্রে,

$$\begin{aligned} P(C = Yes | X) &= \frac{P(X | C=Yes) \times P(C=Yes)}{P(X)} \\ &= \frac{P(\text{Cloudy} | C=Yes) \cdot P(\text{Warm} | C=Yes) \cdot P(\text{Outdoor} | C=Yes) \cdot P(C=Yes)}{P(\text{Cloudy}) \cdot P(\text{Warm}) \cdot P(\text{Outdoor})} \\ &= \frac{\frac{2}{3} \cdot \frac{0}{3} \cdot \frac{2}{3} \cdot \frac{3}{7}}{\frac{3}{7} \cdot \frac{3}{7} \cdot \frac{3}{7}} \\ &= 0 \end{aligned}$$

এখন, অনেকেই হয়তো বুঝতে পারেননি, $P(\text{Cloudy} | C = Yes)$ -এর মান কেন $\frac{2}{3}$ হলো।
ওপরের টেবিল 8.3.1-এ প্রথমেই দেখুন, *Wear Coat*-এর মান *yes* আছে মোট 3 জায়গায়
(শেষের তিন সারিতে)। এই তিন সারিতে দেখুন, *Outlook* কলামে *Cloudy* আছে মোট 2টি
আর *Sunny* আছে 1টি। অর্থাৎ তিনটি $C = Yes$ -এর মধ্যে দুটির জন্য আমরা *Outlook* কলামে
Cloudy পাচ্ছি।

তাই $P(\text{Cloudy} | C = Yes)$ -এর মান দাঁড়ায়, $\frac{\text{মোট কয়টি } \text{Cloudy} \text{ আমরা } C=yes \text{-এর জন্য পেয়েছি}}{\text{মোট কয়টি } \text{Cloudy} \text{ আছে } outlook \text{ কলামে}} = \frac{2}{3}$

এইভাবে বাকিগুলোও হিসাব করা হয়েছে।

এখন, একইভাবে,

$$\begin{aligned} P(C = No | X) &= \frac{P(X | C=No) \times P(C=No)}{P(X)} \\ &= \frac{P(\text{Cloudy} | C=No) \cdot P(\text{Warm} | C=No) \cdot P(\text{Outdoor} | C=No) \cdot P(C=No)}{P(\text{Cloudy}) \cdot P(\text{Warm}) \cdot P(\text{Outdoor})} \\ &= \frac{\frac{1}{4} \cdot \frac{3}{4} \cdot \frac{1}{4} \cdot \frac{4}{7}}{\frac{3}{7} \cdot \frac{3}{7} \cdot \frac{3}{7}} \end{aligned}$$

অধ্যায় ৮ : নাইভ বেইজ ক্লাসিফায়ার (Naïve Bayes Classifier)

$$\begin{aligned} &= \frac{49}{144} \\ &= 0.340 \end{aligned}$$

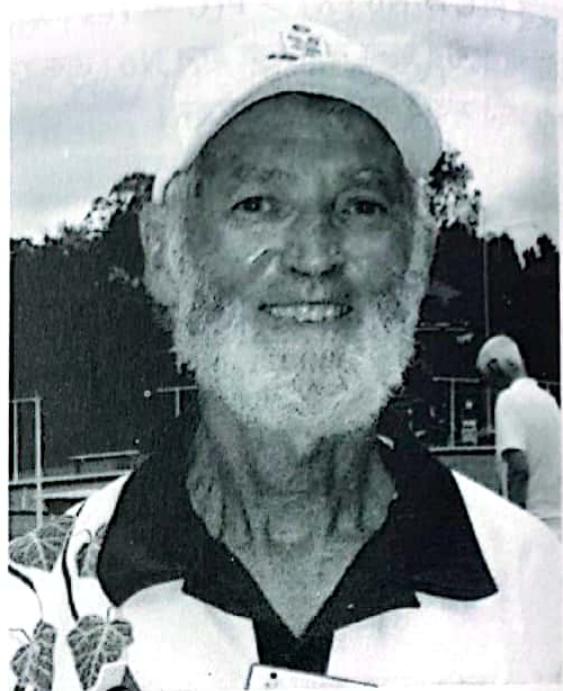
যেহেতু $P(C = \text{No} | X) > P(C = \text{Yes} | X)$, সুতরাং {Cloudy, Warm, Outdoor} এই ডেটা পয়েন্টের জন্য ফলাফল হবে No। এভাবেই একটি নাইভ বেইজ ক্লাসিফায়ার কাজ করে।
আশা করি সবাই বুঝতে পেরেছেন।

অধ্যায় ৯ : ডিসিশন ট্ৰি (Decision Tree)

ডিসিশন ট্ৰি – নাম থেকেই আন্দাজ কৰা যাচ্ছে অনেকটা যে এটি এক ধৰনের ট্ৰি, যেটি কিনা আমাদেৱ কোনো পৰিস্থিতিতে একটি যুক্তিযুক্ত সিদ্ধান্তে পৌছতে সাহায্য কৰে। ট্ৰি তৈৱিৱ ব্যাপারগুলো অনেকে হয়তো আগে থেকে জানেন, যাঁৱা গ্ৰাফ থিওৱি নিয়ে কিছুটা জানেন, আবার পড়াশোনা কৱেছেন। এ ছাড়া সাধাৱণ অ্যালগৱিদম কিংবা ডেটা স্ট্ৰাকচাৱ কোৰ্সেও ট্ৰি সম্পর্কে পড়েছেন অনেকেই আশা কৱি।

আমৱা যে ডিসিশন ট্ৰি তৈৱিৱ অ্যালগৱিদম নিয়ে পড়্ব (ID3, সামনে এটি নিতে বিস্তাৱিত আলোচনা আছে) তাৱ উভাবক রস কুইনল্যান (Ross Quinlan, 1943)।

এটি আমাৱ সবচেয়ে প্ৰিয় মেশিন লাৰ্নিং অ্যালগৱিদম, আৱ তাই এই অধ্যায়টি বেশ বড়োসড়ো হয়ে যাবে। সবাইকে একটু হাত-পা ছড়িয়ে নিয়ে তাৱপৱে অধ্যায়টি পড়তে বসাৱ আমন্ত্ৰণ জানাচ্ছি।



ছবি ৯.১ : Ross Quinlan (1943)

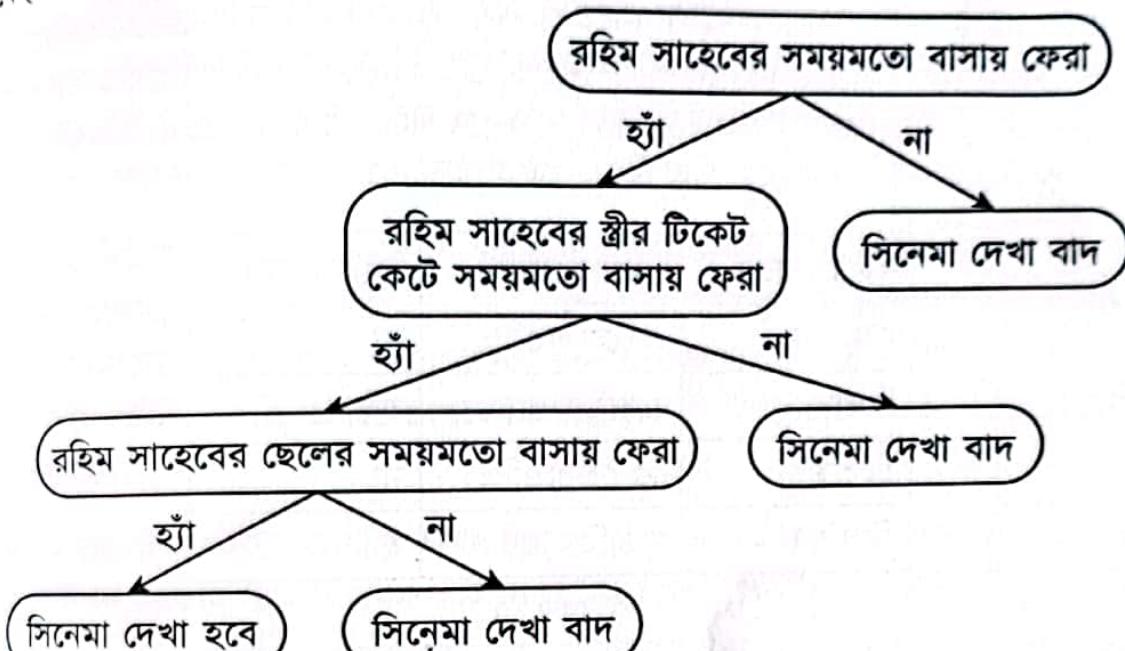
পৱিষ্ঠে ৯.১ : ডিসিশন ট্ৰি কী

ডিসিশন ট্ৰি তৈৱিৱ পদ্ধতিতে যাওয়াৱ আগে, আগে একটু বুৰো নিই ডিসিশন ট্ৰি আসলে কীৱকম। ধৰা যাক, রহিম সাহেব ঠিক কৱেছেন তিনি বাসায় ফিৰে আজকে তাঁৰ পৱিষ্ঠাৱেৱ সবাইকে নিয়ে সিনেমা দেখতে যাবেন। বাসাৱ পাশেই বলাকা সিনেমা হল। যদিও রাস্তাঘাটেৱ যে অবস্থা, তিনি সময়মতো বাসায় ফিৰতে পাৱবেন কি না সেটি এক দুষ্পিত্তা। বাসাৱ কাছাকাছি এসে তাঁৰ মনে পড়ল যে তাঁৰ অনলাইনে টিকিট কাটাৱ কথা ছিল, কিন্তু তিনি কাটতে ভুলে গেছেন। এখন সিনেমা হলে গিয়েই দেখতে হবে টিকিট আছে কি না। তিনি তাঁৰ স্ত্ৰীকে ফোন দিলেন টিকিট কাটতে যাওয়াৱ জন্য, কিন্তু জানতে পাৱলেন তাঁৰ স্ত্ৰী মাৰ্কেটে গেছেন, ফেৱাৱ পথে কেটে আনতে পাৱবেন তিনি টিকিট। তাঁৰ ছেলেকে ফোন দিলেন, ছেলে গেছে কোচিংয়ে। তাঁৰ মানে দাঁড়াচ্ছে, যদি তিনি

অধ্যায় ৯ : ডিসিশন ট্ৰি (Decision Tree)

সময়মতো বাসায় পৌছাতে পারেন, তাঁৰ স্তৰী যদি মার্কেট থেকে টিকিট কিনে সময়মতো বাসায় ফিরতে পারেন, ছেলে যদি সময়মতো কোচিং থেকে বাসায় ফিরতে পারে, তাহলেই কেবল সবাই বেড়ি হয়ে সুন্দরমতো হলে যেতে পারবেন সিনেমা দেখার জন্য, নাহলে পারবেন না।

এখন এই পরিস্থিতির জন্য আমরা যদি একটি ডিসিশন ট্ৰি বানাই, তাহলে সেটি দেখতে এৱেকম হবে (ছবি ৯.১.১) :



ছবি ৯.১.১

এই হচ্ছে মোটামুটি একটি ডিসিশন ট্ৰি-ৰ চেহারা। এই ট্ৰি-ৰ রুট (root) হিসেবে আছে রহিম সাহেবের বাসায় ফেরার ব্যাপারটি।

এখন কথা হচ্ছে, এটাকে রুট হিসেবে না নিয়ে তো আমরা রহিম সাহেবের ছেলের সময়মতো বাসায় ফেরার ব্যাপারটিকেও রুট হিসেবে নিতে পারতাম। কিংবা রহিম সাহেবের স্তৰীর টিকিট কাটার ব্যাপারটিকেও আমরা রুট হিসেবে নিতে পারতাম। তাহলে? কোনটিকে নেব রুট হিসেবে এবং কেন? আৱ ট্ৰি-এৰ যে নোডগুলো ইন্টাৱিমিডিয়েট নোড (যেগুলো লিফ নোড নয়, অর্থাৎ যাদেৱ আৱ কোনো চাইল্ড নোড নেই) তাদেৱ ক্ৰমই বা কীভাৱে ঠিক কৱা যে, কোনটিৰ পৰে কোনটি আসবে?

এটি ঠিক কৱাৱ জন্যই ডিসিশন ট্ৰি তৈৱিতে ইটাৱেটিভ ডাইকটোমাইজিং (Iterative Dichotomiser 3 -ID3) নামে একটি অ্যালগৱিদম ব্যবহাৱ কৱা হয়। ID3 অ্যালগৱিদম শিখতে হলৈ আগে দুটি বিষয় সম্বন্ধে ধাৰণা রাখতে হবে – এন্ট্ৰপি ও গেইন। সেগুলো নিয়ে আমরা এখন জানব।

পরিচ্ছেদ ৯.২ : এন্ট্রপি (Entropy)

এন্ট্রপির সোজাসাপটা বাংলা হচ্ছে বিশৃঙ্খলা। আমরা অনেকেই হয়তো উচ্চমাধ্যমিক পর্যায়ের পদাৰ্থবিজ্ঞানে তাপগতিবিজ্ঞান (Thermodynamics) পড়াৰ সময় এই এন্ট্রপি নিয়ে পড়েছিলাম।

আমাদেৱ এই মেশিন লার্নিংয়ে এন্ট্রপি বলতে বোঝাবে, আমরা আমাদেৱ কোনো ডেটাসেটকে যদি কোনো একটি ফিচারেৰ সাপেক্ষে পার্টিশনিং কৰি, তাহলে সেই পার্টিশনিং কৰে ভালোভাবে আমাদেৱ টার্গেট ভ্যারিয়েবলেৰ কলামকে পার্টিশন কৰতে পাৰবে সেটি। নিচেৱ টেবিলটি (টেবিল 9.2.1) দেখি।

বইয়েৱ ধৰণ	বইয়েৱ লেখক	বইটি কিনব?
ফিকশন	জে কে রাওলিং	না
থিলার	সত্যজিৎ রায়	হ্যাঁ
থিলার	জে কে রাওলিং	না
ফিকশন	সত্যজিৎ রায়	হ্যাঁ

টেবিল 9.2.1

প্ৰথমে বলে নেই, পার্টিশনিং ব্যাপারটি আসলে এৱকম – কোনো একটি ফিচার কলামেৰ যত ভিন্ন মান থাকবে, প্ৰতিটিৰ জন্য আমরা টার্গেট ভ্যারিয়েবলেৰ ভিন্ন ভিন্ন মান নেব।

যেমন, ‘বইয়েৱ লেখক’ – এই কলামে দেখুন, দুটি ভিন্ন ভিন্ন লেখকেৰ নাম আছে, ‘জে কে রাওলিং’ ও ‘সত্যজিৎ রায়’। আমরা যদি এখন, এই ‘বইয়েৱ লেখক’ কলাম দিয়ে পুৱো ডেটাসেটকে পার্টিশন কৰি, তাহলে দুটো পার্টিশন পাৰ। প্ৰথমটি জে কে রাওলিংয়েৰ জন্য, যে ক্ষেত্ৰে টার্গেট কলাম ‘বই কিনব?’ এৱ সবগুলো মানই ‘না’।

বইয়েৱ ধৰণ	বইয়েৱ লেখক	বইটি কিনব?
ফিকশন	জে কে রাওলিং	না
থিলার	জে কে রাওলিং	না

টেবিল 9.2.2

আৱ দ্বিতীয়টি সত্যজিৎ রায়েৰ জন্য, যে ক্ষেত্ৰে টার্গেট কলাম ‘বই কিনব?’ এৱ সবগুলো মানই ‘হ্যাঁ’।

অধ্যায় ৯ : ডিসিশন ট্ৰি (Decision Tree)

বইয়ের ধৰন	বইয়ের লেখক	বইটি কিনব?
ফিকশন	সত্যজিৎ রায়	হ্যাঁ
থ্রিলার	সত্যজিৎ রায়	হ্যাঁ

টেবিল 9.2.3

এখন, আমরা যদি একটু খেয়াল করি, তাহলে দেখব, আমরা ‘বইয়ের লেখক’ দিয়ে পার্টিশনিং কৰার ফলে যে দুটো পার্টিশন তৈরি হলো, তাৰা কিন্তু আমাদেৱ টার্গেট কলামটিকে মিনিমাম বা শূন্য এন্ট্রিপিতে পার্টিশন কৰতে পেৱেছে। অর্থাৎ, প্ৰথম পার্টিশনেৱ জন্য সব টার্গেট ভ্যালুৱ মানই শূন্য, এখানে কোনো ‘হ্যাঁ-না’ৰ মিশ্রণ নেই, অর্থাৎ একটি ‘হ্যাঁ’, আৱেকটি ‘না’ – এৱেকম নয়।

একইভাৱে, দ্বিতীয় পার্টিশনেৱ ক্ষেত্ৰেও সবগুলো টার্গেট ভ্যালু ‘হ্যাঁ’, কোনো ‘হ্যাঁ-না’ৰ মিশ্রণ নেই। যদি মিশ্রণ থাকত, তাহলে এন্ট্রিপি অনেক বেশি হতো, অর্থাৎ বিশৃঙ্খলা বেশি হতো। আমরা সেটি চাই না। আমাদেৱ লক্ষ্যই হলো এন্ট্রিপিৰ মান কমিয়ে নিয়ে আসা। যদি আমরা ‘বইয়ের লেখক’ কলাম দিয়ে পার্টিশন কৰি, তাহলে দেখতেই পাচ্ছি দুটো পার্টিশনেৱ জন্যই আমাদেৱ এন্ট্রিপিৰ মান শূন্য হবে।

এখন, আমরা যদি ‘বইয়ের লেখক’ কলাম দিয়ে পার্টিশন না কৰে ‘বইয়ের ধৰন’ নামেৱ কলাম দিয়ে পার্টিশন কৰতাম, তাহলে প্ৰথম পার্টিশন হতো ‘ফিকশন’ আউটকামেৱ জন্য –

বইয়ের ধৰন	বইয়ের লেখক	বইটি কিনব?
ফিকশন	জে কে রাওলিং	না
ফিকশন	সত্যজিৎ রায়	হ্যাঁ

টেবিল 9.2.4

পৰবৰ্তী পার্টিশন হতো ‘থ্রিলার’ আউটকামেৱ জন্য –

বইয়ের ধৰন	বইয়ের লেখক	বইটি কিনব?
থ্রিলার	সত্যজিৎ রায়	হ্যাঁ
থ্রিলার	জে কে রাওলিং	না

টেবিল 9.2.5

এখন এই ‘বইয়ের ধৰন’ দিয়ে কৱা পার্টিশন দুটি যদি দেখি, তাহলে দেখব যে এখানে এন্ট্রিপি রয়েছে, কাৱণ পার্টিশনেৱ কাৱণে টার্গেট কলামে হ্যাঁ-না মিশ্রণ চলে এসেছে।

অর্থাৎ, আমাদের ডিসিশন ট্রি-তে যখনই আমরা কোনো ডিসিশন নোড বেছে নেব (ক্লট নোড কিংবা অন্য যে-কোনো অন্তর্বর্তী নোড) তখন সেটি বেছে নেওয়ার সময় আমাদের এই এন্ট্রপির আশ্রয় নিতে হবে এবং দেখতে হবে কোনটি ব্যবহার করলে আমাদের সবচেয়ে কম এন্ট্রপি হবে।

এন্ট্রপি গাণিতিকভাবে বের করার একটি সূত্র আছে। সূত্রটি হচ্ছে –

যে-কোনো সেট S-এর জন্য,

$$\text{Entropy}(S) = \sum_{i=1}^n -P_i \log_2 P_i$$

এবং এই এন্ট্রপি হিসাব করতে হবে যে-কোনো পার্টিশনের জন্য মোট কয়টি ভিন্ন ভিন্ন টার্গেট ভ্যারিয়েবল আছে সেখান থেকে।

যেমন, আমাদের ডেটাসেটের জন্য, টার্গেট ভ্যারিয়েবলের মোট চারটি মানের ভেতরে 'না' আছে দুটি, 'হ্যাঁ' আছে দুটি। সুতরাং,

$$\begin{aligned}\text{Entropy}(S) &= \sum_{i=1}^n -P_i \log_2 P_i \\ &= -\left(\frac{2}{4} \log_2 \frac{2}{4} + \frac{2}{4} \log_2 \frac{2}{4}\right) \\ &= 1\end{aligned}$$

এখানে P_i হচ্ছে যে-কোনো একটি টার্গেট ভ্যারিয়েবলের মান ঘটার সম্ভাব্যতা। যেমন, আমাদের ডেটাসেটে 'না' আছে দুটি, মোট ডেটা চারটি। সুতরাং টার্গেট ভ্যারিয়েবল 'না' হওয়ার সম্ভাব্যতা, $\frac{2}{4}$ ।

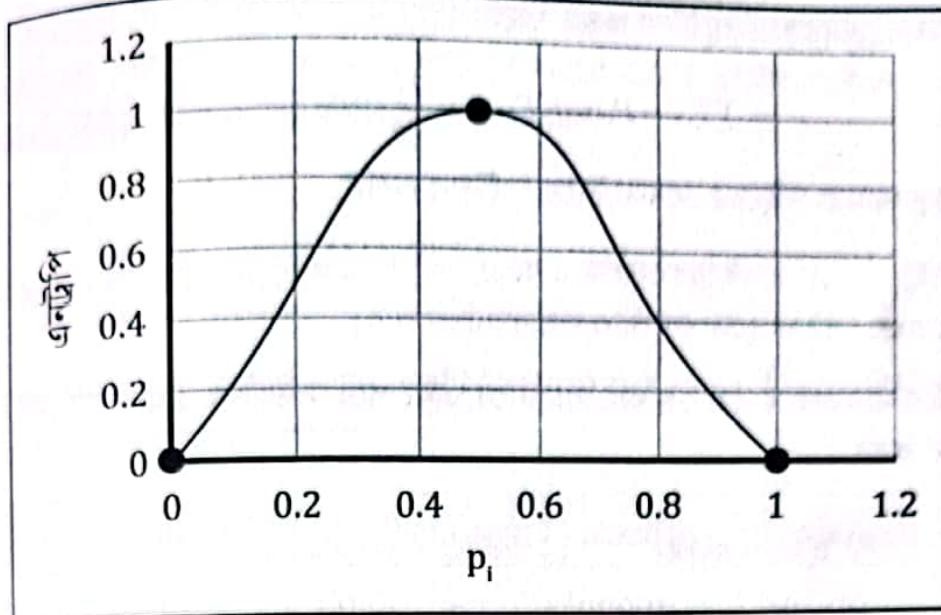
গ্রাফ 9.2.1 লক্ষ করুন। এই গ্রাফটি হচ্ছে, এন্ট্রপি গ্রাফ, যেটি থেকে এন্ট্রপি সমন্বে একটি ধারণা পাওয়া যায়। গ্রাফের X-অক্ষ বরাবর নেওয়া হয়েছে P_i ও Y-অক্ষ বরাবর নেওয়া হয়েছে এন্ট্রপির মান।

গ্রাফ থেকে দেখা যাচ্ছে যে, যদি টার্গেট ভ্যারিয়েবলের আউটকামগুলো কোনো একটি পার্টিশনের জন্য সব এক ধরনের হয় (সবগুলো 'হ্যাঁ' কিংবা সবগুলো 'না') তাহলে সেই পার্টিশনের এন্ট্রপি হয় শূন্য।

আর যদি, সমানসংখ্যক থাকে প্রতিটি আউটকাম (যেমন, ধরা যাক 10টি আউটকামের মধ্যে 5টি হ্যাঁ, 5টি না – এরকম) সে ক্ষেত্রে সম্ভাব্যতা হয় 0.5 এবং এই মানের জন্যই এন্ট্রপি ভ্যালু দেখুন সবচেয়ে বেশি।

আমাদের লক্ষ্যই হচ্ছে এমনভাবে পার্টিশন করা, যাতে এন্ট্রপির মান সর্বনিম্নে অর্থাৎ শূন্যে নামিয়ে আনতে পারি।

অধ্যায় ৯ : ডিসিশন ট্ৰি (Decision Tree)



গ্রাফ 9.2.1

পরিচেদ ৯.৩ : গেইন (Gain)

গেইন মানে হচ্ছে কোনো কিছু পাওয়া, অর্জন করা, তাই না? মেশিন লার্নিংয়ের ক্ষেত্রেও গেইন মানে ঠিক তা-ই। আমরা কোনো গাণিতিক হিসাবনিকাশ করে কিছু একটা পাব, সেটিই হবে আমাদের গেইন। এখন কথা হচ্ছে কী পাব?

আমরা এতক্ষণ এন্ট্রোপি পড়ার সময় বলেছি যে, আমাদের লক্ষ্য হচ্ছে পার্টিশনের পরের এন্ট্রোপি কমানো। অর্থাৎ, যদি আমাদের পার্টিশনের আগের মোট এন্ট্রোপি হয় E_A এবং পরের মোট এন্ট্রোপি হয় E_B তাহলে, আমরা চাইছি E_A ও E_B -এর মধ্যেকার পার্থক্য যতটা সম্ভব বাড়াতে, অর্থাৎ $E_B \ll E_A$ করতে।

এই E_A ও E_B -এর মধ্যেকার মানের পার্থক্যই হচ্ছে গেইন। আমাদের লক্ষ্য হচ্ছে E_B -এর মান যত পারা যায় মিনিমাইজ করার মাধ্যমে গেইনের মান যত পারা যায় ম্যাঞ্জিমাইজ করা।

তাহলে, গেইনের সূত্র হচ্ছে :

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times Entropy(S_v)$$

আমাদের বইয়ের ডেটাসেটের জন্য, আমরা যদি এখন, 'বইয়ের লেখক' দিয়ে পার্টিশন করি, তাহলে,

$Entropy(S)$ = পার্টিশন করার আগের এন্ট্রোপি

$$= \sum_{t=1}^n -P_t \log_2 P_t = -\left(\frac{2}{4} \log_2 \frac{2}{4} + \frac{2}{4} \log_2 \frac{2}{4}\right) = 1$$

এখন, যেহেতু আমরা 'বইয়ের লেখক' দিয়ে পার্টিশন করছি,

তাই, A = বইয়ের লেখক, এবং

$Values(A) = \{\text{জে কে রাওলিং, সত্যজিৎ রায়}\}$

সুতরাং v -এর মান প্রথমে 'জে কে রাওলিং' নিয়ে, এবং পরে 'সত্যজিৎ রায়' নিয়ে সেই পার্টিশনের এন্ট্রোপি বের করব –

$$Entropy(S_{\text{জে কে রাওলিং}}) = -\left(\frac{4}{4} \log_2 \frac{4}{4} + \frac{0}{4} \log_2 \frac{0}{4}\right) = 0$$

$$Entropy(S_{\text{সত্যজিৎ রায়}}) = -\left(\frac{0}{4} \log_2 \frac{0}{4} + \frac{4}{4} \log_2 \frac{4}{4}\right) = 0$$

সবশেষে, 'জে কে রাওলিং' নিয়ে পার্টিশনের ক্ষেত্রে $\frac{|S_v = \text{জে কে রাওলিং}|}{|S|} = \frac{2}{4}$, যেহেতু জে কে রাওলিং নিয়ে মোট চারটি উদাহরণের মধ্যে 2টি উদাহরণ আছে এবং একইভাবে সত্যজিৎ রায় নিয়ে পার্টিশনের ক্ষেত্রে, $\frac{|S_v = \text{সত্যজিৎ রায়}|}{|S|} = \frac{2}{4}$,

সুতরাং সবশেষে,

$$\begin{aligned} Gain(S, A) &= Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times Entropy(S_v) \\ &= 1 - \left(\frac{2}{4} \times 0 + \frac{2}{4} \times 0 \right) \\ &= 1 \end{aligned}$$

একইভাবে, 'বইয়ের ধরন' দিয়ে পার্টিশন করে আমরা গেইন পাই,

$$\begin{aligned} Gain(S, A) &= Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times Entropy(S_v) \\ &= 1 - \left[\frac{2}{4} \times \left\{ -\left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}\right) \right\} + \frac{2}{4} \times \left\{ -\left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}\right) \right\} \right] \\ &= 1 - \left(\frac{2}{4} \times 2 \right) \\ &= 0 \end{aligned}$$

অধ্যায় ৯ : ডিসিশন ট্ৰি (Decision Tree)

আমাদের লক্ষ্য ছিল যে পার্টিশনের গেইন বেশি হবে, আমরা সেই পার্টিশন ব্যবহার করব। আমরা দেখতে পাইছি যে, 'বইয়ের লেখক' দিয়ে পার্টিশন করলে আমরা গেইন সর্বোচ্চ পাইছি, সুতৰাং আমরা পার্টিশন করব 'বইয়ের লেখক' দিয়ে।

পরিচেদ ৯.৪ : কীভাবে ডিসিশন ট্ৰি বানাব

এখন আমরা গোড়া থেকে সম্পূর্ণ নতুন একটি উদাহরণ করে দেখব কীভাবে ডিসিশন ট্ৰি তৈরি কৰতে হয়। আমাদের নতুন ডেটাসেট :

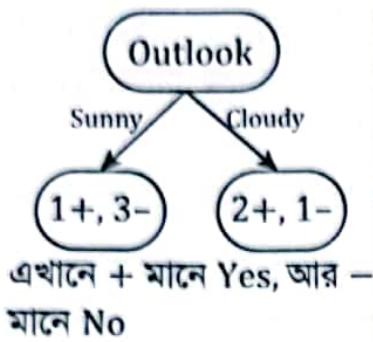
Day	Outlook	Temperature	Routine	Wear Coat?
D_1	Sunny	Cold	Indoor	No
D_2	Sunny	Warm	Outdoor	No
D_3	Cloudy	Warm	Indoor	No
D_4	Sunny	Warm	Indoor	No
D_5	Cloudy	Cold	Indoor	Yes
D_6	Cloudy	Cold	Outdoor	Yes
D_7	Sunny	Cold	Outdoor	Yes

মন করে দেখুন, এই ডেটাসেটটিই আমরা নাইভ বেইজ ক্লাসিফায়ার তৈরি কৰার সময় ব্যবহার কৰেছিলাম। সেটিই আবারও ব্যবহার করে আমরা ডিসিশন ট্ৰি-ও তৈরি কৰব।

Root Selection:

আমাদের সবার আগে ডিসিশন ট্ৰি-এর Root Selection কৰতে হবে। একটু আগে যে পদ্ধতি দিখলাম, সেই পদ্ধতিতে। আমাদের এখানে ফিচার আছে তিনটি – Outlook, Temperature ও Routine। আৱ আমাদের ডেটা আছে মোট ৭টি। এখন আমরা তিনটি ফিচারের প্রতিটির জন্য আলাদা আলাদা কৰে, এদের দিয়ে পুরো ডেটাসেট পার্টিশন কৰলে কত গেইন হবে সেটি বেৱে কৰব। এৱপৰ, যেটি সর্বোচ্চ গেইন দেবে তাকে রুট নোড হিসেবে নিৰ্বাচন কৰব।

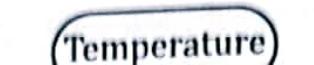
$$\text{এখানে, পার্টিশন কৰার আগের এন্ট্রপি } E(S) = - \left(\frac{3}{7} \log_2 \frac{3}{7} + \frac{4}{7} \log_2 \frac{4}{7} \right) = 0.985$$



$$E(\text{Outlook_Sunny}) = -\left(\frac{1}{4} \log_2 \frac{1}{4} + \frac{3}{4} \log_2 \frac{3}{4}\right) = 0.811$$

$$E(\text{Outlook_Cloudy}) = -\left(\frac{2}{3} \log_2 \frac{2}{3} + \frac{1}{3} \log_2 \frac{1}{3}\right) = 0.918$$

$$\begin{aligned} \text{Gain}(S, \text{Outlook}) &= E(S) - \frac{4}{7} \times 0.811 \\ &\quad - \frac{3}{7} \times 0.918 \\ &= 0.985 - \frac{4}{7} \times 0.811 \\ &\quad - \frac{3}{7} \times 0.918 \\ &= 0.128 \end{aligned}$$

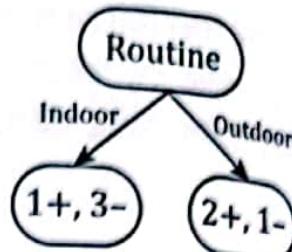


এখনে + মানে Yes, আর - মানে No

$$E(\text{Temperature_Cold}) = -\left(\frac{1}{4} \log_2 \frac{1}{4} + \frac{3}{4} \log_2 \frac{3}{4}\right) = 0.811$$

$$E(\text{Temperature_Warm}) = -\left(\frac{0}{3} \log_2 \frac{0}{3} + \frac{3}{3} \log_2 \frac{3}{3}\right) = 0$$

$$\begin{aligned} \text{Gain}(S, \text{Temperature}) &= E(S) - \frac{4}{7} \times 0.811 \\ &\quad - \frac{3}{7} \times 0 \\ &= 0.985 - \frac{4}{7} \times 0.811 \\ &\quad - \frac{3}{7} \times 0 \\ &= 0.521 \end{aligned}$$



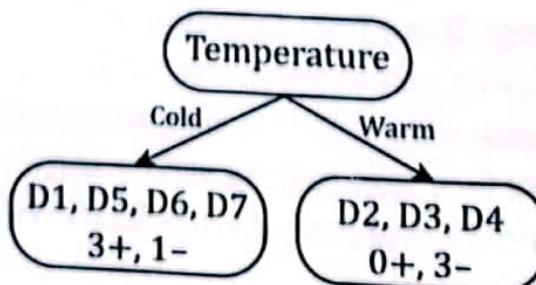
এখনে + মানে Yes, আর - মানে No

$$E(\text{Routine_Indoor}) = -\left(\frac{1}{4} \log_2 \frac{1}{4} + \frac{3}{4} \log_2 \frac{3}{4}\right) = 0.811$$

$$E(\text{Routine_Outdoor}) = -\left(\frac{2}{3} \log_2 \frac{2}{3} + \frac{1}{3} \log_2 \frac{1}{3}\right) = 0.918$$

$$\begin{aligned} \text{Gain}(S, \text{Routine}) &= E(S) - \frac{4}{7} \times 0.811 \\ &\quad - \frac{3}{7} \times 0.918 \\ &= 0.985 - \frac{4}{7} \times 0.811 \\ &\quad - \frac{3}{7} \times 0.918 \\ &= 0.128 \end{aligned}$$

এখন, ওপরের তিনটি গেইন-এর মানের মধ্যে, Temperature-এর মাধ্যমে করা পার্টিশনের গেইন সবচেয়ে বেশি। সুতরাং, আমরা Temperature-কেই আমাদের রুট নোড হিসেবে নির্বাচন করব। তাহলে, আমরা এখন একটি পার্শিয়াল ডিসিশন ট্রি (Partial Decision Tree) পেয়ে গেলাম, যেটি দেখতে এরকম (ছবি 9.4.1) :



ছবি 9.4.1

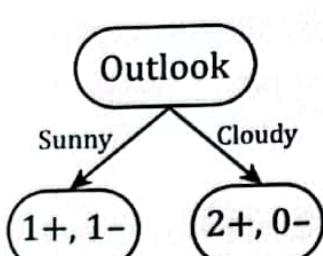
অধ্যায় ৯ : ডিসিশন ট্ৰি (Decision Tree)

এখন, আমাদের ডেটাসেটটি দুটি সাবসেটে ভাগ হয়ে গেল। এর মধ্যে একটি সাবসেটের এন্ট্রপি শূন্য (ডান দিকেরটি), সুতরাং ওটি নিয়ে আমাদের আর কোনো মাথাব্যথা থাকবে না। ওই নোডে পৌছানো মানে *Wear Coat*-এর মান সব সময়েই No হবে।

এখন বাকি থাকে, বাঁ দিকের সাবসেট – D1, D5, D6 এবং D7। এটিকে আমাদের আবার পার্টিশন করতে হবে এবং ততক্ষণ পর্যন্ত এই পার্টিশন করা চালিয়ে যেতে হবে যতক্ষণ পর্যন্ত না আমরা এন্ট্রপি শূন্য পাচ্ছি। আমাদের এই নতুন সাবসেটের নাম দিই S1। এই S1-এর এন্ট্রপি হবে –

$$E(S1) = -\left(\frac{3}{4} \log_2 \frac{3}{4} + \frac{1}{4} \log_2 \frac{1}{4}\right) = 0.811$$

এখন আমরা এই নতুন সাবসেটকে আমাদের মূল ডেটাসেট বিবেচনা করে এর ওপরে আগের মতো পার্টিশন চালিয়ে দেখব কোনটিতে গেইন বেশি পাওয়া যায়। আমরা যেহেতু Temperature ফিচারটি ইতিমধ্যেই ব্যবহার করে ফেলেছি, তাই আমাদের এখন শুধু Outlook ও Routine – এই দুটি ফিচার দিয়ে পার্টিশন করে দেখলেই চলবে।

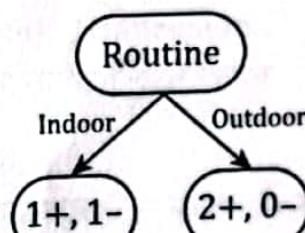


এখানে + মানে Yes, আর – মানে No

$$E(\text{Outlook}_\text{Sunny}) = -\left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}\right) = 1$$

$$E(\text{Outlook}_\text{Cloudy}) = -\left(\frac{2}{2} \log_2 \frac{2}{2} + \frac{0}{2} \log_2 \frac{0}{2}\right) = 0$$

$$\begin{aligned} \text{Gain}(S, \text{Outlook}) &= E(S1) - \frac{2}{4} \times 1 - \frac{2}{4} \times 0 \\ &= 0.811 - \frac{2}{4} \times 1 - \frac{2}{4} \times 0 \\ &= 0.311 \end{aligned}$$



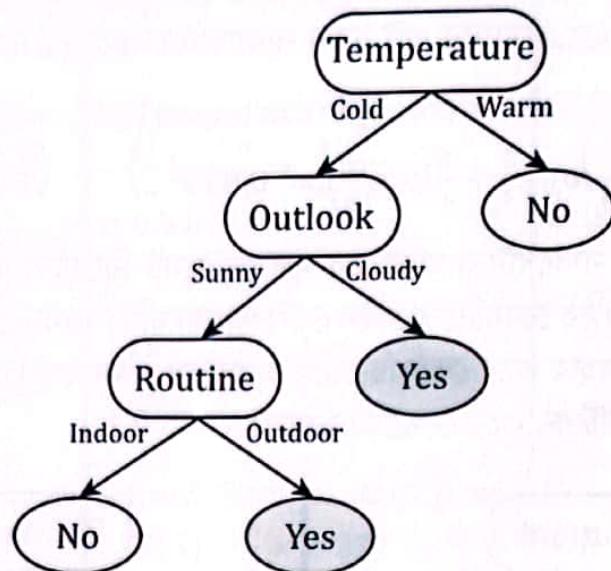
এখানে + মানে Yes, আর – মানে No

$$\begin{aligned} E(\text{Routine}_\text{Indoor}) &= -\left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}\right) = 1 \\ E(\text{Routine}_\text{Outdoor}) &= -\left(\frac{2}{2} \log_2 \frac{2}{2} + \frac{0}{2} \log_2 \frac{0}{2}\right) = 0 \end{aligned}$$

$$\begin{aligned} \text{Gain}(S, \text{Routine}) &= E(S1) - \frac{2}{4} \times 1 - \frac{2}{4} \times 0 \\ &= 0.811 - \frac{2}{4} \times 1 - \frac{2}{4} \times 0 \\ &= 0.311 \end{aligned}$$

মেশিন লার্নিং অ্যালগরিদম

এখন দেখা যাচ্ছে, আমাদের Outlook ও Routine দুটির ক্ষেত্রেই গেইলের পরিমাণ একই।
সুতরাং যে-কোনোটিকেই আমরা পরবর্তী ডিসিশন নোড হিসেবে ধরে নিতে পারি। আমরা যদি
Outlook-কে আমাদের পরবর্তী ডিসিশন নোড ধরে নিই, তাহলে বাকি থাকে শুধু Routine
সেটি দিয়েও আমরা একইভাবেই পার্টিশন করে দেখব। সব ঠিকঠাকমতো করতে পারলে
আমাদের সর্বশেষ ডিসিশন ট্রি দাঁড়াবে এরকম (ছবি 9.4.2) :



ছবি 9.4.2

সুতরাং আমরা শিখে ফেললাম কীভাবে ID3 অ্যালগরিদম প্রয়োগ করে ডিসিশন ট্রি তৈরি করতে
হয়। আশা করি সকলেই বুঝতে পেরেছেন।

অধ্যায় ১০ : প্রিন্সিপাল কম্পোনেন্ট অ্যানালাইসিস (Principal Component Analysis - PCA)

ধূলি, কোন ডেটাসেটের জন্য, ফিচারের সংখ্যা অনেক বেশি ($> 10,000$) এবং সেই তুলনায় ক্রিনিং ডেটা পর্যাপ্ত নেই। এ ধরনের ক্ষেত্রে হিসাবনিকাশ অনেক বেশি জটিল হয়ে যায় এবং Costing-ও অনেক বেড়ে যায় (সময়, মেমোরি বেশি লাগে ইত্যাদি)।

এসব ক্ষেত্রে, এত এত ফিচারের মধ্যে খুব ভালোমতো যদি ডেটা অ্যানালাইসিস করা যায়, তাহলে দেখা যাবে মাত্র কিছুসংখ্যক ফিচার আমাদের দরকার, বাকিগুলো না হলেও হবে। তখন, আমরা অপ্রয়োজনীয় ফিচারগুলো বাদ দিয়ে দিই, যাকে বলে ডাইমেশন রিডাকশন (Dimension Reduction)।

এক ধরনের অ্যালগরিদম আছে, যেগুলো ডেটার এই ডাইমেশন রিডাকশনের কাজ করে দেয়, এগুলোকে বলে ডাইমেনশনালিটি রিডাকশন অ্যালগরিদম (Dimensionality Reduction Algorithm)। এই ধরনের কোনো একটি অ্যালগরিদম ব্যবহার করে আমরা ডেটার ফিচারের সংখ্যা কমিয়ে নিয়ে আসি।

এই ধরনেরই একটি অ্যালগরিদম হলো প্রিন্সিপাল কম্পোনেন্ট অ্যানালাইসিস (Principal Component Analysis) বা পিসিএ (PCA)। নাম থেকেই বোঝা যাচ্ছে, কোথাও অনেকগুলো কম্পোনেন্ট আছে, আমাদের সেখান থেকে যে কম্পোনেন্ট/কম্পোনেন্টগুলো সবচেয়ে বেশি ফুলপূর্ণ সেগুলো রেখে বাকিগুলো বাদ দিয়ে দিতে হবে। ব্যাপারটি আসলেই অনেকটা এরকম। মশিন লার্নিংয়ের ক্ষেত্রে, এই কম্পোনেন্ট বলতে বোঝায় ফিচার ডেটা।

এই বিষয় নিয়ে পড়ার আগে পরিসংখ্যান এবং লিনিয়ার অ্যালজেব্রার কিছু ছোটো ছোটো বিষয় আমরা চট করে দেখে নেব। অনেকেই হয়তো এগুলো জানেন, অনেকেই হয়তো জানেন না। সিখাটি মূলত যাঁরা জানেন না তাঁদের জন্যই, আর যাঁরা জানেন তাঁরাও আরেকবার ঝালাই করে নিলে ক্ষতি কী? আমরা শুরু করব পরিসংখ্যান এর কিছু জনপ্রিয় ধারণা – মিন (Mean), স্ট্যান্ডার্ড ডেভিয়েশন (Standard Deviation) ও ভ্যারিয়েন্স (Variance) দিয়ে, হাতে-কলমে দেখব – কীভাবে এগুলো বের করতে হয়। এরপর আমরা লিনিয়ার অ্যালজেব্রার দুটি বিষয় দেখব – কীভাবে একটি ম্যাট্রিক্সের জন্য আইগেনভ্যালু (Eigenvalue) ও আইগেনভেক্টর (Eigenvector) বের করতে হয়। সবশেষে আমরা, এগুলো ব্যবহার করে কীভাবে পিসিএ অ্যালগরিদম কাজ করে সেটি দেখব।

পরিচ্ছেদ ১০.১ : মিন (Mean), স্ট্যান্ডার্ড ডেভিয়েশন (Standard Deviation) এবং ভ্যারিয়েন্স (Variance)

এগুলো আমরা সবাই-ই ছোটোবেলায় কমবেশি পড়ে এসেছি। তাও, আরেকবার ঝালাই করে নিই। মিন (Mean) হচ্ছে গাণিতিক গড়।

$$\text{গড় বের করার সূত্র}, \bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

এই ভয়াবহ চেহারার সমীকরণের মূল বক্তব্য হলো, যতগুলো ডেটা থাকবে, সবগুলোকে যোগ করে, ডেটার সংখ্যা দিয়ে ভাগ দিলেই গড় পাওয়া যাবে।

কোনো ডেটাসেটের গড় থেকে সেই ডেটাসেটের মানগুলোর ব্যাপারে একটু ধারণা পাওয়া যায়। তবে এর পাশাপাশি, যদি আমরা সেই ডেটাসেটের স্ট্যান্ডার্ড ডেভিয়েশন বের করতে পারি, তাহলে ডেটাসেটের মানগুলোর ব্যাপারে আরো পরিক্ষার ধারণা পাওয়া যায়।

$$\text{স্ট্যান্ডার্ড ডেভিয়েশন বের করার সূত্র}, SD = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{(n-1)}}$$

সহজ করে বলতে গেলে স্ট্যান্ডার্ড ডেভিয়েশন যা বোঝায় তা হলো, আমাদের ডেটাসেটের প্রতিটি ডেটা সেই ডেটাসেটের গড়মান থেকে কত দূরে আছে (কম বা বেশি), সেই দূরত্বের মানগুলোর বর্গের গড় মানের বর্গমূল বা সংক্ষেপে আরএমএস (RMS) গড়।

এখানে, একটি প্রশ্ন প্রায়ই সবার মাথায় আসে যে, আমাদের মোট ডেটার সংখ্যা তো n , তাহলে কেন আমরা n দিয়ে ভাগ না করে $(n-1)$ দিয়ে ভাগ করছি? এর কারণটি ভেঙে বলার চেষ্টা করছি।

ধরা যাক, আপনি পৃথিবীর সব মানুষের বয়সের গড় বের করতে চান। তাহলে আপনাকে কী করতে হবে? পৃথিবীর সব মানুষের বয়স এক এক করে যোগ করে তাকে মোট মানুষের সংখ্যা দিয়ে ভাগ করতে হবে, তাই না? কিন্তু, বাস্তবে এই কাজটি করা অসম্ভব; কোনোভাবেই প্রতিটি মানুষের কাছে গিয়ে, তার বয়স জেনে নিয়ে সব যোগ করে গড় বের করা সম্ভব নয়।

তাই এ ক্ষেত্রে যাঁরা এই ধরনের ডেটা নিয়ে কাজ করতে যান, তাঁরা যেটি করেন, তা হলো, এই সাড়ে সাত বিলিয়ন মানুষের ডেটাসেট না বানিয়ে আরো ছোটো সাইজের ডেটাসেট (ধরুন, 1000 জন মানুষের ডেটাসেট) তৈরি করেন। তাঁরা এই 1000 জন মানুষকে এমনভাবে নির্বাচন করেন যাতে, এতে পৃথিবীর ভিন্ন ভিন্ন প্রায় সব দেশের, সব বয়সের, ধর্মের, বর্ণের, গোত্রের মানুষ থাকে। কিংবা তাঁরা যে ডোমেইন নিয়ে কাজ করতে চান, সে ডোমেইনের সব ধরনের ভ্যারিয়েশন যেন এই ডেটাসেটে বিদ্যমান থাকে, সেটি নিশ্চিত করেন।

অধ্যায় ১০ : প্রিন্সিপাল কম্পোনেন্ট অ্যানালাইসিস (Principal Component Analysis - PCA)

হ্রন্ত, তাঁরা যদি বিভিন্ন দেশের মানুষের মধ্যে পর্যালোচনা করতে চান, তাঁরা নিশ্চিত করেন বা হ্রন্তে চান যে তাঁদের ডেটাসেটে যেন প্রতিটি দেশ থেকে একজন মানুষের তথ্য থাকে। এইভাবে ধরে নেওয়া হয় যে, ওই একজন মানুষের সম্পর্কে তথ্য পেলে সেটি ওই গোটা দেশের মানুষের সম্পর্কেই তথ্য পেয়ে যাওয়া হবে। এই ধরনের ডেটাসেটকে বলা হয় স্যাম্পল ডেটা সেট এবং সেটি আমাদের প্রধান সেট, যেটি পৃথিবীর সব মানুষকে নিয়ে যে সেট তৈরি হবে, তার একটি স্বাক্ষর সেট।

এখন ধরা যাক, আপনি যদি আপনার প্রধান ডেটাসেট নিয়ে কাজ করতেন, তাহলে তার স্ট্যান্ডার্ড ডেভিয়েশন হতো S এবং স্যাম্পল ডেটাসেট নিয়ে কাজ করার কারণে যদি আপনি n দিয়ে ভাগ করেন, তাহলে আপনার স্ট্যান্ডার্ড ডেভিয়েশন হবে S_n , এবং যদি আপনি $n - 1$ দিয়ে ভাগ করেন, তাহলে সেটি হবে S_{n-1} ।

বাস্তবে দেখা যায়, প্রধান ডেটাসেটের বদলে স্যাম্পল ডেটাসেট ব্যবহার করা হলে, S_n -এর চেয়ে S_{n-1} -এর মান প্রধান ডেটাসেটের S -এর বেশি কাছাকাছি হয়। আর আমরা যেহেতু বেশিরভাগ সময়ই প্রধান ডেটাসেটের বদলে স্যাম্পল ডেটাসেট ব্যবহার করি, তাই আমরা n -এর পরিবর্তে $n - 1$ দিয়ে ভাগ করি।

যদি, আমরা স্যাম্পল ডেটাসেট ব্যবহার না করে প্রধান ডেটাসেটই ব্যবহার করতাম, তাহলে সময়ই আমরা n দিয়ে ভাগ করতাম।

স্বাক্ষরের যে সূত্রটি, সেটি হলো ভ্যারিয়েন্স (Variance)-এর সূত্র। ভ্যারিয়েন্স ও স্ট্যান্ডার্ড ডেভিয়েশন প্রায় একই জিনিস। স্ট্যান্ডার্ড ডেভিয়েশনের মানকে বর্গ করলেই আমরা ভ্যারিয়েন্স পেয়ে যাই।

$$\text{সূত্রাঃ, } Var(X) = SD^2 = \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{(n-1)}$$

পরিচ্ছেদ ১০.২ : কোভ্যারিয়েন্স (Covariance) ও কোভ্যারিয়েন্স ম্যাট্রিক্স (Covariance Matrix)

কোভ্যারিয়েন্স (Covariance) ও ভ্যারিয়েন্সের মধ্যে অনেকখানিই মিল আছে, পার্থক্য শুধু ভ্যারিয়েন্স কাজ করে একমাত্রিক বা 1-D ডেটা দিয়ে, আর কোভ্যারিয়েন্স কাজ করে দ্বিমাত্রিক বা 2-D ডেটা দিয়ে। উদাহরণ দিয়ে বোঝানোর চেষ্টা করছি।

আমাদের সেই পিংজার উদাহরণটিতেই ফিরে যাই :

মেশিন লার্নিং অ্যালগরিদম

পিংজার সাইজ (ইঞ্জিতে)	পিংজার দাম (টাকায়)
6"	350/-
8"	775/-
12"	1150/-
14"	1395/-
18"	1675/-

টেবিল 10.2.1

এখানে দেখুন, পিংজার সাইজ হচ্ছে একটি ডাইমেনশন এবং পিংজার দাম হচ্ছে আরেকটি ডাইমেনশন (যদিও পিংজার দামকে আসলে ঠিক ডেটার ডাইমেনশন হিসেবে সরাসরি ধরা যাবে না, তবু এখানে ধরে নিচ্ছি বোঝানোর সুবিধার্থে)।

এখন আমরা যদি শুধু পিংজার সাইজ কিংবা পিংজার দাম নিয়ে কাজ করতাম, তাহলে আমরা ব্যবহার করতাম ভ্যারিয়েন্স। কিন্তু যদি আমরা বের করতে যাই যে, পিংজার সাইজ বাড়ার সঙ্গে কি পিংজার দাম বাড়ছে, নাকি কমছে, ইত্যাদি, অর্থাৎ দুটি ভিন্ন ভিন্ন ডাইমেনশনের ডেটার মধ্যেকার সম্পর্কটি আসলে কী, সেটি যদি বুঝতে চাই, তাহলে আমাদের ব্যবহার করতে হবে কোভ্যারিয়েন্স।

কোভ্যারিয়েন্সকে লেখা হয় এভাবে – $Cov(X, Y)$; এখানে X হচ্ছে প্রথম ডাইমেনশনের ডেটাসেট, Y হচ্ছে দ্বিতীয় ডাইমেনশনের ডেটাসেট। এটি বের করার সূত্রও অনেকটাই ভ্যারিয়েন্সের সূত্রের মতো –

$$Cov(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$

এখন, $Cov(X, Y)$ -এর মান যা-ই আসুক, সেটি আমাদের মাথাব্যথা নয়। আমাদের শুধু দেখতে হবে যে, মানটি ধনাত্মক, শূন্য নাকি ঋণাত্মক।

- যদি মান ধনাত্মক হয়, তাহলে আমরা বলতে পারব যে ডাইমেনশন দুটি ধনাত্মক সম্পর্কযুক্ত বা পজিটিভলি কোরিলেটেড (Positively Correlated) অর্থাৎ, একটির মান বাড়লে অন্যটির মান বাড়বে, একটির মান কমলে অন্যটি কমবে।
- আবার, মান যদি ঋণাত্মক হয়, তাহলে আমরা বলতে পারি যে এরা ঋণাত্মক সম্পর্কযুক্ত বা নেগেটিভলি কোরিলেটেড (Negatively Correlated) অর্থাৎ একটির মান বাড়লে অন্যটির মান কমবে।

অধ্যায় ১০ : প্রিসিপাল কম্পোনেন্ট অ্যানালাইসিস (Principal Component Analysis - PCA)

- সবশেষে, যদি মান শূন্য আসে, তাহলে বুঝে নিতে হবে যে এই দুটি ডাইমেনশনের মধ্যে আসলে কোনো ধরনের সম্পর্ক নেই, তারা পরস্পর স্বাধীন বা ইনডিপেনডেন্ট (Independent)।

আরেকটি বিষয় মনে রাখতে হবে, যদি আমরা X ডাইমেনশনের সঙ্গে এর নিজেরই কোভ্যারিয়েন্স হিসাব করি, অর্থাৎ যদি $Cov(X, X)$ নিই, তাহলে আমরা এই X ডাইমেনশন বরাবর ভ্যারিয়েন্স পেয়ে যাব।

এখন আসি কোভ্যারিয়েন্স ম্যাট্রিক্স (Covariance Matrix)-এ। আমরা ইতিমধ্যেই জা যে, কোভ্যারিয়েন্স ম্যাট্রিক্স শুধু দুটি ডাইমেনশনের মধ্যেকার সম্পর্ক নিয়ে কাজ করে। এখন, ধরা যাব, আমাদের ডাইমেনশন আছে তিনটি কিংবা তারও বেশি। তখন কী করা? তখন সবগুলো ডাইমেনশনের মধ্যে থেকে দুটি দুটি করে নিয়ে আমাদেরকে তাদের কোভ্যারিয়েন্স বের করতে হব এবং সেগুলো নিয়ে কাজ করতে হবে। যখন, এমন অবস্থা থাকবে, তখন আসলে কোভ্যারিয়েন্স ম্যাট্রিক্স তৈরি করে নিতে হয় এবং তাতে প্রতি জোড়া ডাইমেনশনের মধ্যেকার কোভ্যারিয়েন্সের মান রাখতে হয়।

যখন, ধরা যাক, আমাদের ডাইমেনশন এখন তিনটি – X , Y এবং Z । তাহলে আমাদের কোভ্যারিয়েন্স ম্যাট্রিক্স হবে এরকম –

$$C = \begin{pmatrix} Cov(X, X) & Cov(X, Y) & Cov(X, Z) \\ Cov(Y, X) & Cov(Y, Y) & Cov(Y, Z) \\ Cov(Z, X) & Cov(Z, Y) & Cov(Z, Z) \end{pmatrix}$$

এখন উল্লেখ্য যে $Cov(X, Y)$ ও $Cov(Y, X)$ আসলে একই মান।

পরিচ্ছদ ১০.৩ : আইগেনভ্যালু (Eigenvalue) ও আইগেনভেক্টর (Eigenvectors)

পরিশেষে, পিসিএ পড়ার আগে আমাদের শেষ আরেকটি বিষয় একটু জানতে হবে এবং এটি ইচ্ছ সবচেয়ে গুরুত্বপূর্ণ বিষয়। এটি ভালোভাবে বুঝতে হবে। এটি হলো আইগেনভেক্টর (Eigenvector) ও আইগেনভ্যালু (Eigenvalues)। আগে খুব সহজ করে ধারণাটি দিই, এপরে কীভাবে এ দুটি হিসাব করতে হয় তা বর্ণনা করছি।

নিচের ম্যাট্রিক্স গুণনাটি দেখি –

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

এখানে আমরা যেটি করলাম, তা হলো একটি 2×2 ম্যাট্রিক্সকে একটি কলাম ভেষ্টের দিয়ে শুণ করলাম, করে আমরা $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ পেলাম। এতটুকু পর্যন্ত খুব সহজ, সাধারণ ম্যাট্রিক্স গুণনের সূত্র। পরবর্তী সময়ে দেখুন, এই ফলাফলটিকে আমরা আবার আমাদের প্রথম যেই কলাম ভেষ্টের ছিল $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$ সেটির একটি পূর্ণ সংখ্যার গুণিতক আকারে লিখতে পারলাম, অর্থাৎ এই পুরোনো কলাম ভেষ্টেরটিকে একটি ইন্টিজার দিয়ে শুণ করেই আমরা আমাদের ফলাফল $\begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}$ পেয়ে যাচ্ছি। এরকম ক্ষেত্রে, $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$ -কে বলা হবে এই $\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}$ ম্যাট্রিক্সের একটি আইগেনভেষ্টের এবং 4 -কে বলা হবে $\begin{pmatrix} 2 & 3 \\ 1 & 1 \end{pmatrix}$ ম্যাট্রিক্সের একটি আইগেনভ্যালু।

আইগেনভেষ্টের কিছু গুরুত্বপূর্ণ ধর্ম –

- আইগেনভেষ্টের সব সময় $n \times n$ অর্থাৎ বর্গ বা ক্ষয়ার ম্যাট্রিক্সের থাকে।
- সব ক্ষয়ার ম্যাট্রিক্সের আইগেনভেষ্টের থাকে না।
- কোনো $n \times n$ ম্যাট্রিক্সের যদি আইগেনভেষ্টের থাকে, তাহলে আইগেনভেষ্টেরের সংখ্যা হবে n -সংখ্যক, অর্থাৎ যদি 3×3 ম্যাট্রিক্স হয়, তাহলে তার 3টি আইগেনভেষ্টের থাকতে পারে।
- প্রতিটি আইগেনভেষ্টের একে অপরের ওপরে লম্ব (Perpendicular)।
- প্রতিটি আইগেনভেষ্টেরের সঙ্গে একটিমাত্র আইগেনভ্যালু জড়িত থাকবে।

এই গেল মোটামুটি আইগেনভেষ্টের ও আইগেনভ্যালু সম্পর্কে একটি প্রাথমিক ধারণা। এগুলো লিনিয়ার অ্যালজেব্রা কোর্সে আরো বিষদভাবে জানতে পারা যাবে, যেটি আপাতত আমাদের এই বইয়ে সংযুক্ত করা হচ্ছে না।

এখন, কীভাবে আইগেনভ্যালু ও আইগেনভেষ্টের বের করতে হয় সেটি একটু হাতে-কলমে দেখা যাক। একটি কথা মনে রাখতে হবে যে, 2×2 কিংবা 3×3 ম্যাট্রিক্সের আইগেনভেষ্টের ও আইগেনভ্যালু হয়তো হাতে-কলমে কিংবা কোড করে বের করা সম্ভব, কিন্তু এর চেয়ে উচ্চতর ডাইমেনশনের ম্যাট্রিক্সের আইগেনভ্যালু ও আইগেনভেষ্টের বের করতে হলে অবশ্যই উচিত হবে কোনো সফটওয়্যার লাইব্রেরি কিংবা প্যাকেজ ব্যবহার করা। কেননা এগুলো হাতে-কলমে বের করা একেবারেই অপ্রয়োজনীয় এবং কষ্টসাধ্যও বটে। আমি শুধু বোৰ্ডের সুবিধার্থে এখানে একটি 2×2 ম্যাট্রিক্সের আইগেনভেষ্টের ও আইগেনভ্যালু বের করে দেখাচ্ছি।

ধরা যাক, আমরা নিচের ম্যাট্রিক্সের আইগেনভেষ্টের ও আইগেনভ্যালু বের করব –

$$A = \begin{pmatrix} 7 & 3 \\ 3 & -1 \end{pmatrix}$$

অধ্যায় ১০ : প্রিসিপাল কম্পোনেন্ট অ্যানালাইসিস (Principal Component Analysis - PCA)

গ্রহণে ধরে নিতে হবে যে আইগেনভ্যালু হচ্ছে λ এবং একটি 2×2 আইডেন্টিটি ম্যাট্রিক্স (Identity Matrix) দিয়ে গুণ করতে হবে, যেহেতু আমাদের A ম্যাট্রিক্সটি 2×2 সাইজের। তাহলে আমরা পাই -

$$\lambda I = \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}$$

এরপরে আমাদের বের করতে হবে,

$$A - \lambda I = \begin{pmatrix} 7 & 3 \\ 3 & -1 \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} = \begin{pmatrix} 7-\lambda & 3 \\ 3 & -1-\lambda \end{pmatrix}$$

তৃতীয় ধাপে এসে, আমাদের বের করতে হবে $\det(A - \lambda I)$, অর্থাৎ $\begin{pmatrix} 7-\lambda & 3 \\ 3 & -1-\lambda \end{pmatrix}$ এর নির্ণয়ক। এটি আমরা সবাইই করেছি ছোটোবেলায়। নির্ণয়কটি দাঁড়ায়,

$$(7-\lambda)(-1-\lambda) - 3 \cdot 3 = \lambda^2 - 6\lambda - 16$$

আমাদের এখন $\lambda^2 - 6\lambda - 16 = 0$ সমীকরণটি সমাধান করতে হবে। সমাধান করে পাই,

$$\lambda = 8, -2$$

এই ৮ এবং -2 ই হচ্ছে আমাদের 2×2 ম্যাট্রিক্সের আইগেনভ্যালু। এখন শুধু আইগেনভ্যেটের বের করা বাকি।

$$\lambda = 8 \text{ ধরে } \begin{pmatrix} 7-\lambda & 3 \\ 3 & -1-\lambda \end{pmatrix} = \begin{pmatrix} 7-8 & 3 \\ 3 & -1-8 \end{pmatrix} = \begin{pmatrix} -1 & 3 \\ 3 & -9 \end{pmatrix}$$

এখন ধরি, $X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ হচ্ছে আমাদের আইগেনভ্যেটের, যার জন্য আইগেনভ্যালু হচ্ছে ৮। সুতরাং, এখন আমরা $\begin{pmatrix} -1 & 3 \\ 3 & -9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ এই সমীকরণটি সমাধান করলেই X এর মান পেয়ে যাব।

এখন থেকে আমরা পাই দুটি সমীকরণ -

$$\begin{pmatrix} -1 & 3 \\ 3 & -9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$-x_1 + 3x_2 = 0$$

$$3x_1 - 9x_2 = 0$$

এখন এই দুটি সমীকরণ সমাধান করে আমরা খুব সহজেই পাই, $x_1 = 3$ এবং $x_2 = 1$ ।

সুতরাং, প্রথম আইগেনভ্যালু ৮-এর জন্য আইগেনভ্যেটের হচ্ছে $\begin{pmatrix} 3 \\ 1 \end{pmatrix}$ ।

একই পদ্ধতিতে আমরা আইগেনভ্যালু -2-এর জন্য সমাধান করে পাই, আইগেনভ্যেটের হচ্ছে $\begin{pmatrix} 1 \\ -3 \end{pmatrix}$ ।

অঙ্গে আমরা পেয়ে গেলাম আমাদের $\begin{pmatrix} 7 & 3 \\ 3 & -1 \end{pmatrix}$ ম্যাট্রিক্সের দুটি আইগেনভ্যেটের এবং আইগেনভ্যালু। এরপর আমরা সরাসরি চলে যাব আমাদের পিসিএ-তে।

পরিচ্ছেদ ১০.৪ : কীভাবে প্রিসিপাল কম্পোনেন্ট অ্যানালাইসিস করতে হয়

এবাব আমরা দেখব কীভাবে ধাপে ধাপে পিসিএ প্রয়োগ করে, ডেটাসেটের ডাইমেনশন কমিয়ে আনা যায়।

ধরা যাক, আমাদের নতুন ডেটাসেট এরকম :

X	Y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2.0	1.6
1.0	1.1
1.5	1.6
1.1	0.9

টেবিল 10.4.1

ডেটাসেটের দুটি ডাইমেনশন হচ্ছে X ও Y। আমরা এখন নিচের ধাপগুলো অনুসরণ করব :

- প্রথমেই আমরা যেটি করব, সেটি হচ্ছে ডেটা অ্যাডজাস্টমেন্ট। এই ধাপে, আমরা মূলত প্রতিটি ডাইমেনশনের সব ডেটা থেকে ওই ডাইমেনশনের গড়মান বিয়োগ করে দেব। অর্থাৎ, X-এর প্রতিটি মান থেকে এদের গড়মান 1.81 বিয়োগ করে দেব, একইভাবে Y-এর প্রতিটি মান থেকে এদের গড়মান 1.91 বিয়োগ করে দেব। এর ফলে, X ও Y ডাইমেনশনের ডেটার গড়মান 0 হয়ে যাবে।

এখন তাহলে, নতুন ডেটাসেট দাঁড়াবে এরকম :

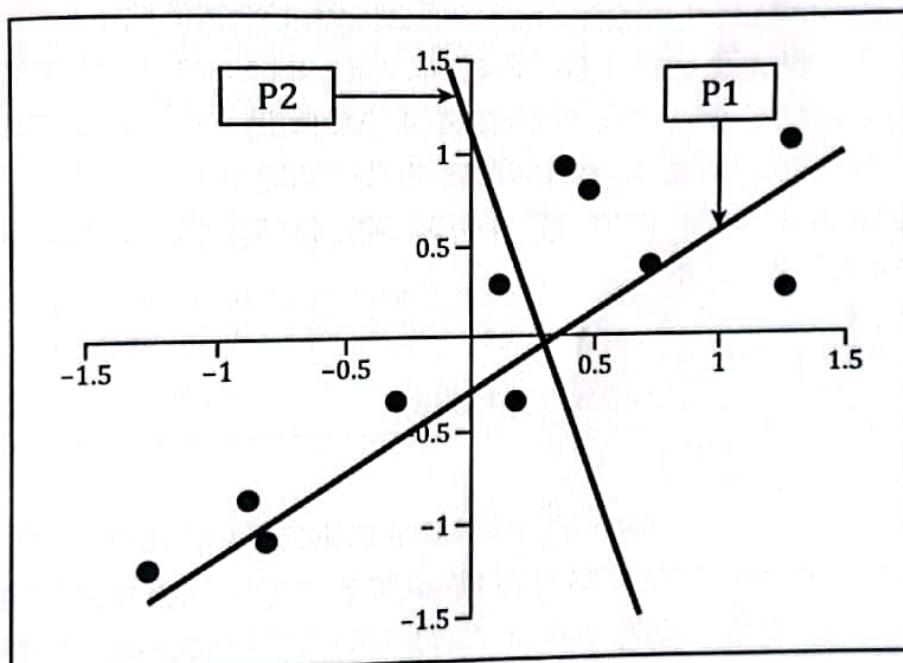
X	Y
0.69	0.49
-1.31	-1.21
0.39	0.99

অধ্যায় ১০ : প্রিমিপাল কম্পোনেন্ট অ্যানালাইসিস (Principal Component Analysis - PCA)

0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

টেবিল 10.4.2

মানগুলোকে গ্রাফ 10.4.1-এ দেখানো হলো :



গ্রাফ 10.4.1

- এর পরের ধাপে আমরা এই ডেটাসেটের জন্য কোভ্যারিয়েন্স ম্যাট্রিক্স তৈরি করব। কোভ্যারিয়েন্স ম্যাট্রিক্স কীভাবে তৈরি করতে হয়, সে সম্পর্কে আমরা আগেই বলেছি। আমাদের এই ডেটাসেটের ক্ষেত্রে, কোভ্যারিয়েন্স ম্যাট্রিক্সটি হবে –

$$Cov = \begin{pmatrix} 0.616555556 & 0.615444444 \\ 0.615444444 & 0.716555556 \end{pmatrix}$$

এই কোভ্যারিয়েন্স ম্যাট্রিক্স থেকে আমরা দেখতে পাচ্ছি যে, $Cov(X, Y)$ -এর মান ধনাত্মক। সুতরাং বলা যায় X -এর মান বাড়ার সঙ্গে সঙ্গে Y -এর মান বাড়ছে।

- এর পরের ধাপে আমাদের এই কোভ্যারিয়েল ম্যাট্রিক্সের আইগেনভেস্টের ও আইগেনভ্যালু বের করতে হবে। আমরা ইতিমধ্যেই দেখে ফেলেছি কীভাবে আইগেনভেস্টের ও আইগেনভ্যালু বের করতে হয়। খুব ভালো হয় যদি আপনারা হাতে-কলমে বা কোড করে এগুলো বের না করে সরাসরি লিনিয়ার অ্যালজেবরার কোনো লাইব্রেরি বা মডিউল ব্যবহার করেন। তাহলে মানগুলো একেবারে নিখুঁত আসবে। তবে চাইলে হাত পাকানোর জন্য পরীক্ষামূলকভাবে হাতে-কলমে করে দেখতে পারেন।

আমি এখানে সরাসরিই লিখে দিচ্ছি মানগুলো –

$$\text{আইগেনভ্যালু, } \begin{pmatrix} 0.490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{আইগেনভেস্টের, } \begin{pmatrix} -0.735178656 & -0.677873399 \\ 0.677873399 & -0.735178656 \end{pmatrix}$$

এখানে বলে রাখা ভালো, আইগেনভেস্টের ম্যাট্রিক্সের প্রথম কলামটি হচ্ছে প্রথম আইগেনভেস্টের এবং দ্বিতীয় কলামটি হচ্ছে দ্বিতীয় আইগেনভেস্টের। একইভাবে, আইগেনভ্যালু ম্যাট্রিক্সের প্রথম মানটি হচ্ছে প্রথম আইগেনভেস্টের আইগেনভ্যালু এবং আইগেনভ্যালু ম্যাট্রিক্সের দ্বিতীয় মানটি হচ্ছে দ্বিতীয় আইগেনভেস্টের আইগেনভ্যালু।

- গ্রাফ 10.4.1-এ খেয়াল করলে, দুটি বাঁকানো অক্ষ দেখতে পাবেন, P1 ও P2। P1 হচ্ছে আমাদের আইগেনভেস্টের –

$$\begin{pmatrix} -0.677873399 \\ -0.735178656 \end{pmatrix}$$

এবং P2 হচ্ছে আমাদের অপর আইগেনভেস্টের –

$$\begin{pmatrix} -0.735178656 \\ 0.677873399 \end{pmatrix}$$

আরো লক্ষ করবেন, P1 অক্ষ বরাবর আমাদের ডেটাসেটের মানগুলো বেশি ছড়ানো, কিন্তু সেই তুলনায় P2 অক্ষ বরাবর আমাদের মানগুলো তুলনামূলকভাবে অনেক কম ছড়ানো।

- এই ব্যাপারটি আইগেনভ্যালু থেকেও বোঝা যায়। যে আইগেনভ্যালুর মান বড়ো, সেই আইগেনভ্যালুর সঙ্গে সম্পৃক্ত আইগেনভেস্টের অক্ষ বরাবর ডেটা বেশি ছড়ানো থাকবে।
- আইগেনভ্যালু দুটির মধ্যে, যেটির মান বড়ো (এখানে দ্বিতীয়টি), সেই আইগেনভ্যালুটি নিয়েই আমরা কাজ করব। যদি আরেকটু জেনারালাইজ করে বলি, তাহলে আমরা যতগুলো আইগেনভ্যালু পাব, সেগুলোকে বড়ো থেকে ছোটো এই ক্রমে সাজাতে হবে। এরপর যদি ইচ্ছা হয়, তাহলে আমরা এখান থেকে যেই আইগেনভ্যালুগুলো কম গুরুত্বপূর্ণ অর্থাৎ যাদের মান ছোটো, চাইলে সেগুলো বাদ দিয়ে বড়ো মানবিশিষ্ট আইগেনভ্যালুগুলো শুধু রেখে দিতে পারি। এতে যদিও আমাদের কিছু তথ্য হারিয়ে যাবে বা ইনফরমেশন লস (Information Loss) হবে, কিন্তু আমাদের ডেটার ডাইমেনশন কমে আসবে এবং শুধু সেই গুরুত্বপূর্ণ ডাইমেনশনের ডেটাই থাকবে যেগুলো ওই ডেটাসেটকে উপস্থাপন করার জন্য যথেষ্ট।

অধ্যায় ১০ : প্রিসিপাল কম্পোনেন্ট অ্যানালাইসিস (Principal Component Analysis - PCA)

আমাদের এই ক্ষেত্রে আমরা ধরে নিচ্ছ যে, আমরা দ্বিতীয় আইগেনভ্যালুটি রেখে দিয়ে প্রথমটি বাদ দিয়ে দেব।

- এখন আমাদের একটি ফিচার ভেস্টের তৈরি করতে হবে, যেটি আসলে আমরা যে-যে আইগেনভ্যালু নিয়ে কাজ করব বলে সিদ্ধান্ত নিয়েছি, সেসব আইগেনভ্যালুর সঙ্গে যেসব আইগেনভেস্টের জড়িত, সেগুলো নিয়ে তৈরি করা একটি ভেস্টের। আমাদের এই ক্ষেত্রে আমরা শুধু দ্বিতীয় আইগেনভেস্টেরটি নিয়েই আমাদের ফিচার ভেস্টের তৈরি করব। তাহলে আমাদের ফিচার ভেস্টেরটি হবে –

$$(-0.677873399) \\ (-0.735178656)$$

- এর পরে আমাদের শেষ ধাপ। সেটি হচ্ছে, আমাদের এখন নতুন ডেটা পেতে হবে যে ডেটার ডাইমেনশন আমাদের অরিজিনাল ডেটাসেটের চেয়ে কম হবে। আমাদের কিন্তু মূল লক্ষ্যই ছিল এটি যে, আমরা আমাদের ডেটাসেটের ডেটার ডাইমেনশন কমিয়ে নিয়ে আসব, যাতে আমাদের কম্পিউটেশনাল কমপ্লেক্সিটি (Computational Complexity) কমে আসে।

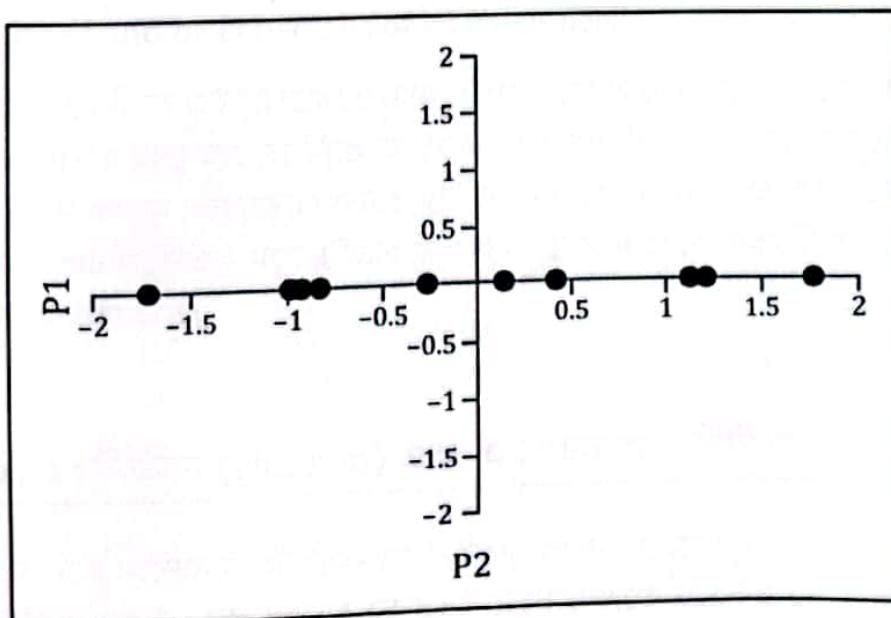
নতুন ডেটা পাওয়ার সূত্র হচ্ছে –

$$\text{Final Data} = (\text{Feature Vector})^T \times (\text{Data Adjust})^T$$

$$= (-0.677873399 \quad -0.735178656) \times$$

$$(0.69 \quad -1.31 \quad 0.39 \quad 0.09 \quad 1.29 \quad 0.49 \quad 0.19 \quad -0.81 \quad -0.31 \quad -0.71) \\ (0.49 \quad -1.21 \quad 0.99 \quad 0.29 \quad 1.09 \quad 0.79 \quad -3.1 \quad -0.81 \quad -0.31 \quad -1.01)$$

$$= (-0.827 \quad 1.777 \quad -0.992 \quad -0.274 \quad -1.675 \quad -0.912 \quad 0.099 \quad 1.144 \quad 0.438 \quad 1.223)$$



গ্রাফ 10.4.2

মেশিন লার্নিং অ্যালগরিদম

এখানে যেটি করা হলো যে, আমাদের ডেটাসেটকে শুধু P1 অক্ষের ওপরে প্রোজেক্ট (Project) করা হলো এবং এর P2 অক্ষ বরাবর ওই ডাইমেনশনে যাবতীয় যা ডেটা ছিল, সব মুছে ফেলা হলো।

আমরা যদি এখন আমাদের P1 অক্ষকে একটু রোটেট (Rotate) করে X-অক্ষের ওপর প্রতিস্থাপন করি এবং সেইসঙ্গে P2 অক্ষকে Y-অক্ষের ওপর প্রতিস্থাপন করি, তাহলে আমরা আমাদের সর্বশেষ ডেটা পাই এরকম (গ্রাফ 10.4.2)।

এই ডেটাই আমাদের নতুন ডেটাসেট যার ডাইমেনশন একটিই। এই ডেটাসেট দিয়েই প্রবর্তী সময়ে আমাদের যাবতীয় মেশিন লার্নিংয়ের কাজ করতে হবে।

এখানে আরেকটি বিষয় উল্লেখ্য যে, পিসিএ আমরা ডেটার ডাইমেনশন কমানোর কাজে ব্যবহার করা ছাড়াও ফিচার এক্সট্রাকশন (Feature Extraction)-এর কাজেও ব্যবহার করতে পারি। তার বিস্তারিত আলোচনায় গেলাম না, কিন্তু জেনে রাখাটা ভালো।

অধ্যায় ১১: পারসেপট্রন (Perceptron)

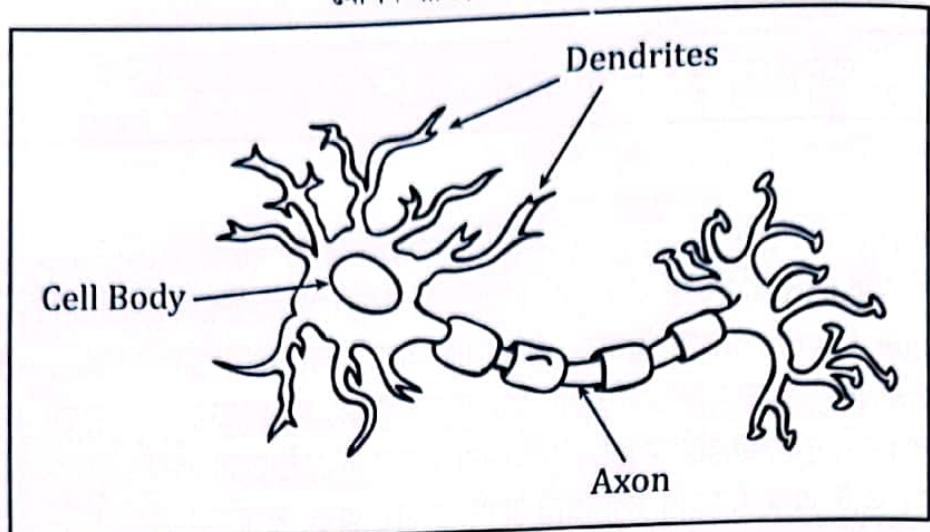
পারসেপট্রন (Perceptron) – নাম শুনেই মনে হচ্ছে বেশ ভারিকি কিছু একটা, তাই না? আসলেই, এটিকে বেশ রাজকীয় অ্যালগরিদমই বলা চলে। পারসেপট্রন (সাধারণত পারসেপট্রন বলতে Single Layer পারসেপট্রনকে বোবায়) একটি বাইনারি ক্লাসিফায়ার অ্যালগরিদম। বাইনারি ক্লাসিফায়ার মানে হচ্ছে, এটি নির্ধারণ করতে পারে যে, কোনো বস্তু কোনো একটি নির্দিষ্ট ক্লাসের কি না। যদি দুইয়ের অধিক ক্লাস থাকে, সে ক্ষেত্রে এই সিংগেল লেয়ার পারসেপট্রন ব্যবহার করা হয় না। এটি এক ধরনের লিনিয়ার ক্লাসিফায়ার এবং সুপারভাইজড লার্নিংয়ের মাধ্যমে ব্যবহৃত হয়ে থাকে।

এই পারসেপট্রন প্রথম উভাবন করা হয় 1957 সালে কর্নেল অ্যারোনাটিক্যাল ল্যাবরেটরি (Cornell Aeronautical Laboratory)-তে। এর উভাবক ছিলেন ফ্র্যাংক রোসেনব্লাট (Frank Rosenblatt, 1928-1971)। একটি খুব মজার খবর বলি, 1957 সালে যখন এই পারসেপট্রন প্রথম ব্যবহৃত হয়, তখন থেকেই ধীরে ধীরে কম্পিউটার ও মেশিন লার্নিংয়ের ভবিষ্যৎ, কম্পিউটারের দ্বারা মানুষের পরাজিত হয়ে বিলুপ্ত হয়ে যাওয়ার আশঙ্কা ইত্যাদি ধারণা মাথাচাড়া দিয়ে উঠতে থাকে। সেসবের ওপরে ভিত্তি করেই, সেই সময়ে ‘ন্য নিউ ইয়র্ক টাইমস’ (The New York Times) পত্রিকায় পারসেপট্রন সম্পর্কে লেখা হয় – ‘the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.’

পরবর্তী সময়ে ধীরে ধীরে এই সিংগেল লেয়ার পারসেপট্রন থেকেই মাল্টি লেয়ার পারসেপট্রন বা নিউরাল নেটওয়ার্কের জন্ম হয়, যা নিয়ে আজকের এই পৃথিবীতে এত মাতামাতি। পারসেপট্রন নিয়ে পড়তে গেলে প্রথমে আমাদের জানতে হবে নিউরন (Neuron) কী এবং এর আদলে মিল রেখে কীভাবে একটি গাণিতিক মডেল তৈরি করা যায়। সেখান থেকে পরবর্তী সময়ে পারসেপট্রন সম্পর্কে ধারণা দেওয়া হবে।

পরিচ্ছেদ ১১.১: নিউরন (Neuron) এবং এর আদলে গাণিতিক কাঠামো

আমরা স্কুল-কলেজে অনেকেই জীববিজ্ঞানে নিউরন সম্পর্কে জেনেছি। নিউরন হচ্ছে আমাদের মস্তিষ্কের গাঠনিক একক, এক ধরনের কোষ। নিউরন দেখতে কীরকম সে সম্পর্কে মোটামুটি একটি ধারণা পাওয়া যাবে নিচের ছবিটি থেকে (ছবি 11.1.1) :



ছবি 11.1.1

একটি নিউরনের মোট তিনটি প্রধান অংশ থাকে, এর কোষদেহ (Cell Body), ডেনড্রাইট (Dendrite) নামের ছোটো ছোটো শাখা-প্রশাখা, যা কোষদেহ থেকে বের হয়েছে এবং থাকে একটি লম্বা দণ্ডের মতো অংশ – অ্যাক্সন (Axon)। এই অ্যাক্সনগুলো আবার পরবর্তী নিউরনের ডেনড্রাইটের সঙ্গে আল্টেপ্রেস্ট সংযুক্ত থাকে, এই সংযোগকে বলা হয় সিন্যাপস (Synapse)। এই সিন্যাপস সংযোগগুলোর কারণেই আমরা চিন্তা করতে পারি, সূতি ধারণ করতে পারি, সিদ্ধান্ত নিতে পারি। নিউরনের ক্ষেত্রে ডেনড্রাইটগুলোকে বলা যেতে পারি ইনপুট ঢোকার পথ, আর অ্যাক্সনকে বলতে পারি আউটপুট বের হবার পথ।

এখন, আমাদের মেশিন লার্নিং পড়ার সময় এই জীববিজ্ঞানে পড়ে আসা নিউরনের কাঠামোর সঙ্গে সামঞ্জস্যপূর্ণ একটি গাণিতিক কাঠামো সম্পর্কে পড়তে হবে, এর নামও নিউরন, শুধু পার্থক্য এই যে এটি গণিতে ব্যবহার করা হয় এবং এর ইনপুট, আউটপুটগুলো হয় বিভিন্ন সংখ্যা।

আমরা এবারে গাণিতিক নিউরনের একটি ছবি দেখে ফেলি (ছবি 11.1.2) :

এখানে $x_1, x_2 \dots x_n$ ইত্যাদি হলো একটি ট্রেনিং ডেটা পয়েন্টের বিভিন্ন ফিচারের মান। এখানে ডেটার ডাইমেনশন n ।

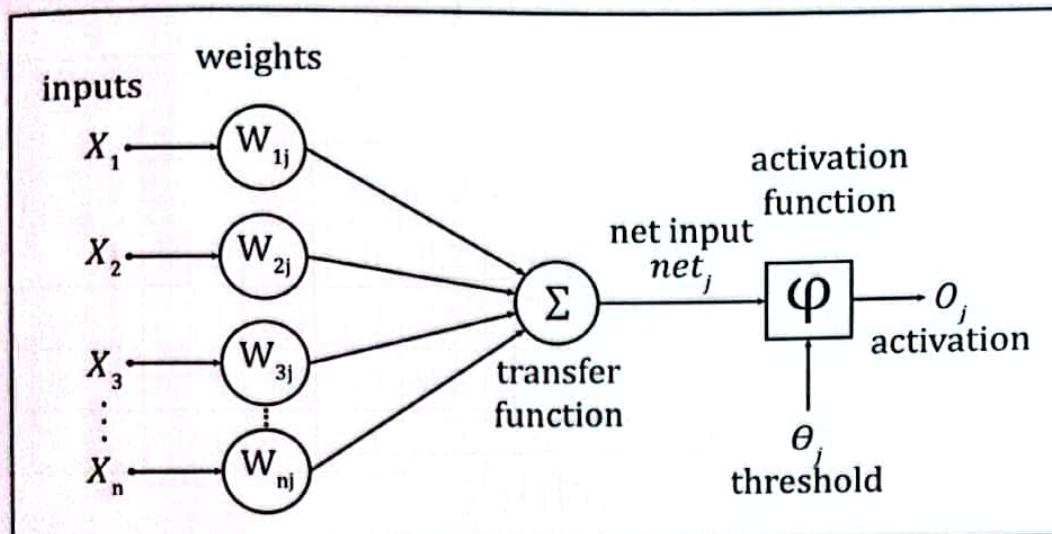
J দিয়ে J -তম ডেটা পয়েন্ট বোঝানো হচ্ছে। আর w হচ্ছে স্বভাবতই আমাদের প্রতিটি ডেটা পয়েন্টের সঙ্গে সম্পৃক্ত ওয়েইট-এর মান।

সহজ ভাষায় এখন বলে দিই পারসেপ্ট্রন কীভাবে কাজ করবে –

- প্রথমে প্রতিটি ইনপুটের সঙ্গে তার ওয়েইট গুণ হয়ে সবগুলো একসঙ্গে যোগ করা হবে। এই কাজটি করবে ট্রান্সফার ফাংশন (Transfer Function) নামে একটি ফাংশন, যেটি দেখানো হয়েছে Σ চিহ্ন দিয়ে। এটি একটি অ্যাডার জাংশন (Adder Junction) বা সামিং জাংশন

অধ্যায় ১১ : পারসেপ্ট্রন (Perceptron)

(Summing Junction)। এর কাজ হচ্ছে, যা ইনপুট পাবে সব যোগ করবে। এর ভেতরে ইনপুট হিসেবে দেওয়া হবে $x_1w_1, x_2w_2 \dots x_nw_n$ ইত্যাদি। বরাবরের মতোই আমরা এই ওয়েইটগুলোতে প্রথমে র্যান্ডম মান বসাব। এই সামিং জাংশন সবগুলোকে যোগ করে আউটপুট দেবে $x_1w_1 + x_2w_2 + \dots + x_nw_n$ । একেই আমরা নিট ইনপুট (net input) net_j , হিসেবে চিহ্নিত করেছি।



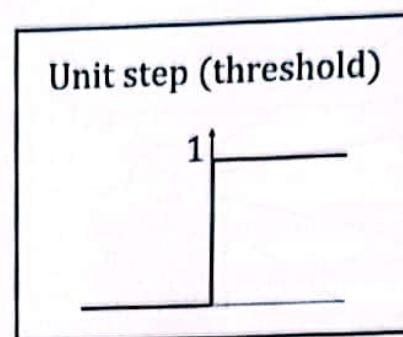
ছবি 11.1.2

- পরবর্তী সময়ে এই net_j -কে একটি থ্রেসহোল্ড (Threshold) মানের সঙ্গে তুলনা করা হবে। এই তুলনা করার কাজটি করবে অ্যাক্টিভেশন ফাংশন (Activation Function) নামে একটি ফাংশন। এই থ্রেসহোল্ড মানকে আমরা আপাতত T_h দিয়ে চিহ্নিত করতে পারি।
এখানে উল্লেখ্য যে, আমাদের পারসেপ্ট্রন কিন্তু শুধুই একটি লিনিয়ার ক্লাসিফায়ার ছাড়া আর কিছুই নয়। যদিও সাধারণত, অ্যাক্টিভেশন ফাংশনগুলো সাধারণত কোনো লিনিয়ার ফাংশনকে নন-লিনিয়ারিটি দেয়, কিন্তু, পারসেপ্ট্রনের ক্ষেত্রে সেটি হয় না (এর কারণ হিসেবে বলা যায়, পারসেপ্ট্রনের ইনপুট এবং আউটপুট ছাড়া মধ্যবর্তী আর কোনো গোপন বা হিডেন (hidden) হিডেন লেয়ার নেই, এ ব্যাপারটি পরবর্তী অধ্যায়ে আলোচনা করা হয়েছে)।
- যদি, net_j অর্থাৎ $\sum_{i=1}^n x_i w_i \geq T_h$ হয়, তাহলে আমাদের পারসেপ্ট্রন আউটপুট দেবে 1, নাহলে আউটপুট দেবে 0। এ ধরনের ফাংশনকে ইউনিট স্টেপ ফাংশন (Unit Step Function)-ও বলা হয়।
ছবি 11.1.3-তে একটি ইউনিট স্টেপ ফাংশন দেখানো হয়েছে।
- ধরা যাক, পারসেপ্ট্রন আউটপুট দিল Y_p আর ডেটা পয়েন্ট X -এর জন্য প্রকৃত আউটপুট হওয়ার কথা Y । এখন, যদি $Y = Y_p$ হয়, তার মানে কোনো এরর হয়নি, সুতরাং আমাদের আর ওয়েইটের মান আপডেট করা লাগবে না। কিন্তু যদি $Y = Y_p$ না হয়, তখন আমাদের ওয়েইটের মানগুলো আপডেট করতে হবে।

ওয়েইটের মান আপডেট করার সূত্র হলো –

$$W_i(j+1) = W_i(j) + \alpha \cdot x_i \cdot (Y - Y_p)$$

এখন, $W_i(j+1)$ মানে হচ্ছে নতুন ওয়েইট, $W_i(j)$ হচ্ছে পুরোনো ওয়েইট, α হচ্ছে আমাদের সেই লার্নিং রেট, যা আমরা প্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদমে পড়েছিলাম, x_i হচ্ছে ওয়েইট $W_i(j)$ যে ইনপুটের সঙ্গে সম্পর্কযুক্ত (অর্থাৎ, i -তম ট্রেনিং ডেটা) এবং $(Y - Y_p)$ হচ্ছে আমাদের এরর।



ছবি 11.1.3

পরিচেদ 11.2 : পারসেপ্ট্রন ট্রেনিং দেওয়া

এখন আমরা সরাসরি কীভাবে একটি পারসেপ্ট্রনকে ট্রেনিং দিতে হয় সেটি দেখব। আমাদের এই কাজের জন্য ডেটাসেটটি নিম্নরূপ :

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

টেবিল 11.2.1

ওপরের টেবিল দেখে অনেকেই হয়তো বুঝে গেছেন, এটি দুটি ইনপুটের জন্য লজিক্যাল অ্যান্ড (Logical AND) অপারেশনের ট্রুথ টেবিল (Truth Table)। এটিই হবে আমাদের ইনপুট, আমরা চাই আমাদের এই পারসেপ্ট্রন $(0,0)$ ইনপুট পেলে 0 আউটপুট দেবে; $(1,1)$ ইনপুট পেলে 1 আউটপুট দেবে ইত্যাদি।

অধ্যায় ১১ : পারসেপট্রন (Perceptron)

এখন তাহলে আমাদের ট্রেনিং শুরু করা যাক।

আমরা আমাদের এই উদাহরণের জন্য ধরে নিচ্ছি আমাদের প্রাথমিক ওয়েইট হচ্ছে 0.3 এবং -0.1 । তবে কেউ চাইলে অন্য কোনো মানও ধরে নিতে পারেন, শূন্য ধরে নেওয়াটা সবচেয়ে ভালো বুদ্ধি। এই উদাহরণের জন্য $\alpha = 0.1, T_h = 0.2$ ।

আমাদের ট্রেনিং কোনো রাউন্ডে (epoch) সব এররের মান শূন্য না হওয়া পর্যন্ত চলতে থাকবে।

Epoch	Inputs		Y	Initial Weights		Y _p	Error (Y - Y _p)	Final Weights	
	x ₁	x ₂		w ₁	w ₂			w ₁	w ₂
1	0	0	0	0.3	-0.1	0	0	0.3	-0.1
	0	1	0	0.3	-0.1	0	0	0.3	-0.1
	1	0	0	0.3	-0.1	1	-1	0.2	-0.1
	1	1	1	0.2	-0.1	0	1	0.3	0.0

টেবিল 11.2.2

প্রথম রাউন্ডে দেখুন, $(1,0)$ ইনপুটের জন্য আউটপুট হওয়ার কথা 0 , কিন্তু আউটপুট হলো 1 ,

$$\text{সুতরাং} \quad \text{error} = -1$$

আউটপুট 1 হওয়ার কারণ,

$$x_1 w_1 + x_2 w_2 = 1 \times 0.3 + 0 \times (-0.1) = 0.2$$

আমরা বলেছিলাম, যদি $x_1 w_1 + x_2 w_2 \geq T_h$ হয়, তাহলে আমাদের পারসেপট্রন আউটপুট দেবে 1 , আর নাহলে আউটপুট দেবে 0 । যেহেতু আমাদের $T_h = 0.2$ ছিল এবং তা $x_1 w_1 + x_2 w_2 = 0.2$ রাশিটির সমান, তাই পারসেপট্রনের আউটপুট হলো 1 ।

$$\begin{aligned} \text{প্রথম ওয়েইট আপডেট হয়ে দাঁড়াল, } & 0.3 + [0.1 \times 1 \times (-1)] = 0.2 \text{ এবং} \\ \text{দ্বিতীয় ওয়েইট আপডেট হয়ে দাঁড়াল, } & -0.1 + [0.1 \times 0 \times (-1)] = -0.1 \end{aligned}$$

এভাবেই বাকি হিসাবনিকাশ করতে হবে পরের রাউন্ডগুলোর জন্য। এই রাউন্ডেই হিসাব থেমে যাবে না, কেননা আমাদের পারসেপট্রন এই রাউন্ডে একবার ভুল করেছে। যতক্ষণ পর্যন্ত না এমন কোনো রাউন্ড পাওয়া যাবে, যেখানে এই পারসেপট্রন কোনো এরর দেবে না, ততক্ষণ পর্যন্ত আমাদের ট্রেনিং চালিয়ে যেতে হবে।

পরের রাউন্ডের হিসাবনিকাশগুলো দেখে নেওয়া যাক,

দ্বিতীয় রাউন্ড :

Epoch	Inputs		Y	Initial Weights		Y_p	Error ($Y - Y_p$)	Final Weights	
	x_1	x_2		w_1	w_2			w_1	w_2
2	0	0	0	0.3	0.0	0	0	0.3	0.0
	0	1	0	0.3	0.0	0	0	0.3	0.0
	1	0	0	0.3	0.0	1	-1	0.2	0.0
	1	1	1	0.2	0.0	1	0	0.2	0.0

টেবিল 11.2.3

এই রাউন্ডেও দেখুন একবার ভুল আছে, সুতরাং আবারও আমাদের ট্রেনিং দিতে হবে। এখানে উল্লেখ্য, এই রাউন্ডের প্রাথমিক ওয়েইট 0.3 এবং 0.0 আমরা পেয়েছি আগের রাউন্ডের সর্বশেষ ফাইনাল ওয়েইট থেকে। আরেকটি ব্যপার, এপক (Epoch) মানে সহজ ভাষায় রাউন্ড নম্বর বোঝাচ্ছে।

তৃতীয়, চতুর্থ ও পঞ্চম রাউন্ড :

Epoch	Inputs		Y	Initial Weights		Y_p	Error ($Y - Y_p$)	Final Weights	
	x_1	x_2		w_1	w_2			w_1	w_2
3	0	0	0	0.2	0.0	0	0	0.2	0.0
	0	1	0	0.2	0.0	0	0	0.2	0.0
	1	0	0	0.2	0.0	1	-1	0.1	0.0
	1	1	1	0.1	0.0	0	1	0.2	0.1

Epoch	Inputs		Y	Initial Weights		Y_p	Error ($Y - Y_p$)	Final Weights	
	x_1	x_2		w_1	w_2			w_1	w_2
4	0	0	0	0.2	0.1	0	0	0.2	0.1
	0	1	0	0.2	0.1	0	0	0.2	0.1
	1	0	0	0.2	0.1	1	-1	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

অধ্যায় ১১ : পারসেপট্রন (Perceptron)

Epoch	Inputs		Y	Initial Weights		Y _p	Error (Y - Y _p)	Final Weights	
	x ₁	x ₂		w ₁	w ₂			w ₁	w ₂
5	0	0	0	0.1	0.1	0	0	0.1	0.1
	0	1	0	0.1	0.1	0	0	0.1	0.1
	1	0	0	0.1	0.1	0	0	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

টেবিল 11.2.3

যাক, অবশ্যে এই পঞ্চম রাউন্ডে এসে আমরা দেখলাম যে আমাদের পারসেপট্রন আর কোনো ভুল করেনি। তাই, আমাদের পারসেপট্রনকে ট্রেনিং দেওয়া এই রাউন্ডেই সমাপ্ত হবে। চাইলে একে আরো কয়েক রাউন্ড ট্রেনিং দেওয়া যায়, কিন্তু তাতে আমাদের কস্ট (cost) খুব একটা কমবে না।

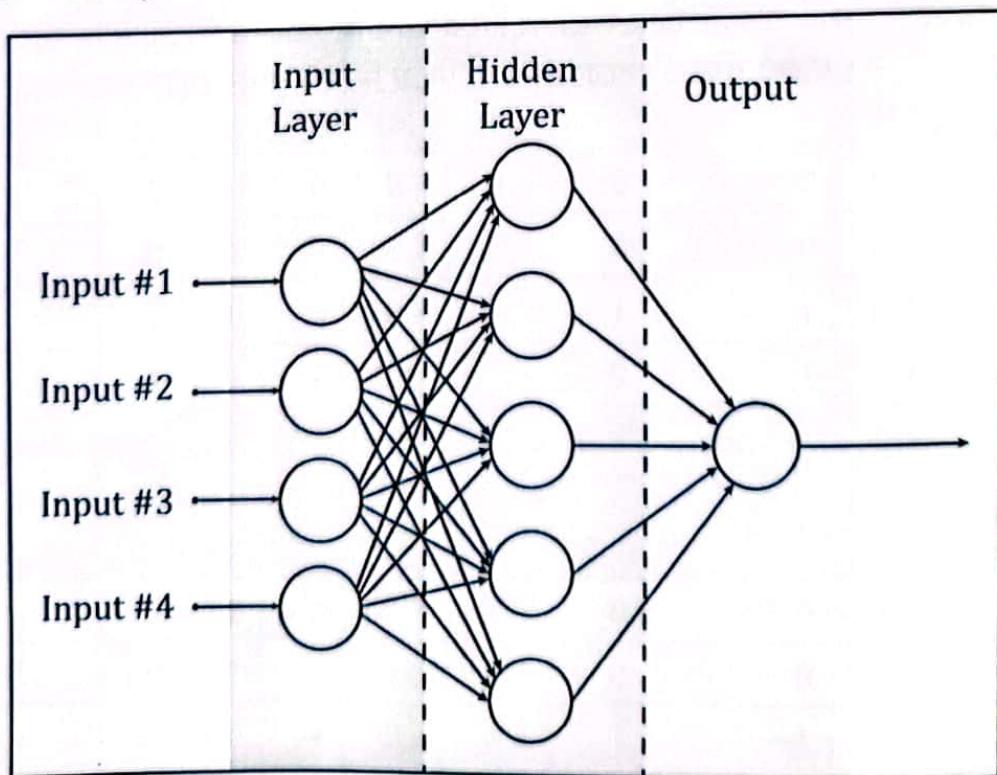
এখানে কিন্তু আমরা স্টকাস্টিক গ্রেডিয়েন্ট ডিসেন্ট (Stochastic Gradient Descent) পদ্ধতি প্রয়োগ করেছি, বুঝতেই পারছেন। গ্রেডিয়েন্ট ডিসেন্ট কীভাবে কাজ করে, এটি যাঁরা ইতিমধ্যেই কিছুটা ভুলে গেছেন, তাঁরা পরিচ্ছেদ ৩.৮ আরেকবার পড়ে নিতে পারেন।

এই ছিল কীভাবে একটি পারসেপট্রনকে ট্রেনিং দিতে হয় সেই সংক্রান্ত আলোচনা। আশা করি, সবাই পারসেপট্রনের ধারণা বুঝতে পেরেছেন এবং নিজেই এখন কোড করে ফেলতে পারবেন।

অধ্যায় ১২ : একটুখানি নিউরাল নেটওয়ার্ক

এবার আসি নিউরাল নেটওয়ার্ক কী, সেই আলোচনায়। একটু ভালো করে দেখলেই বোৰা যাবে যে নামের মধ্যেই এর অর্থ অনেকখানি লুকিয়ে আছে। নেটওয়ার্ক বলতে মূলত বোৰায় কতগুলো জিনিসের একটি সমষ্টি, যেখানে প্রত্যেকের সঙ্গে কোনো-না-কোনোভাবে সংযুক্ত আনেকগুলো কম্পিউটার একে অপরের সঙ্গে বিভিন্ন ধরনের তার (যেমন LAN-এর তার হতে অনেকগুলো কম্পিউটার একে অপরের সঙ্গে বিভিন্ন ধরনের তার (যেমন LAN-এর তার হতে পারে) এবং বিভিন্ন ডিভাইস (যেমন হাব, রাউটার, সুইচ ইত্যাদি) দিয়ে যুক্ত থাকে এবং একটি কম্পিউটার আরেকটি কম্পিউটারের সঙ্গে তথ্য আদান-প্রদান করতে পারে।

নিউরাল নেটওয়ার্ক হচ্ছে নিউরনের নেটওয়ার্ক (একটু আগে আমরা যে নিউরনের ব্যাপারে পড়লাম), যেখানে নিউরনগুলো একে অপরের সঙ্গে যুক্ত থাকে এবং একে অপরের সঙ্গে তথ্য আদান-প্রদান করতে পারে। নিউরনগুলো এক বা একাধিক লেয়ারে সাজানো থাকে। তথ্য হিসাবনিকাশ করা হয় লেয়ার অনুযায়ী (প্রতি লেয়ারে আলাদা হিসাব) এবং তথ্য আদান-প্রদান হয় এক লেয়ার থেকে আরেক লেয়ারে। একটি খুব সাধারণ নিউরাল নেটওয়ার্কের ছবি নিচে দেওয়া হলো (ছবি 12.1.1) :



ছবি 12.1.1

এই ছবিতে একটি তিন লেয়ারবিশিষ্ট নিউরাল নেটওয়ার্ক দেখানো হয়েছে। প্রথম লেয়ারটি হচ্ছে ইনপুট লেয়ার, যেখানে সমস্ত ইনপুট দেওয়া হয়। সর্বশেষ লেয়ারটি হচ্ছে আউটপুট লেয়ার, যেখানে আউটপুট দেওয়া হয়। আর মাঝখানেরটি হচ্ছে হিডেন লেয়ার। এখানে যাবতীয় কঠিন কঠিন হিসাবনিকাশ করা হয়। আরো জটিল নিউরাল নেটওয়ার্কের ক্ষেত্রে হিডেন লেয়ারের সংখ্যা অনেক বেড়ে যায়, হিসাবনিকাশও তখন কঠিন হয়ে যায়।

নিউরাল নেটওয়ার্ক অনেক ধরনের হয়। সবচেয়ে জনপ্রিয় ও বহুল ব্যবহৃত নিউরাল নেটওয়ার্কটি হলো ফিড ফরওয়ার্ড ব্যাকপ্রোপাগেশন নিউরাল নেটওয়ার্ক (Feed Forward Backpropagation Neural Network)। বিশাল খানদানি এক নাম, দেখেই বোঝা যাচ্ছে এটি বেশ জবরদস্ত কিসিমের নিউরাল নেটওয়ার্ক। নিউরাল নেটওয়ার্কগুলোর মধ্যে কিছু ভাগ আছে। নাম দেখে তায় পাওয়ার কিছু নেই।

ফিড ফরওয়ার্ড বলতে মূলত বোঝায় ফরওয়ার্ড প্রোপাগেশন (Forward Propagation), অর্থাৎ সামনের দিকে কোনো কিছু পাঠানো। এটি এক ধরনের অ্যালগরিদম। আবার ব্যাকপ্রোপাগেশন মানে হলো পেছনের দিকে কিছু পাঠানো, তাই না? ঠিক তাই। এটিও এক ধরনের অ্যালগরিদম। সাধারণত এই দুই ধরনের অ্যালগরিদম ব্যবহার করে একটি নিউরাল নেটওয়ার্ককে ট্রেইনিং দেওয়া হয়।

আগের অধ্যায়ে আমরা দেখেছি কীভাবে পারসেপ্ট্রনকে ট্রেইন করাতে হয়, একইভাবে এই নিউরাল নেটওয়ার্ককেও আমাদের ট্রেইন করাতে হবে।

ট্রেনিংয়ের দুটি ধাপ, প্রথম ধাপ হলো ফরওয়ার্ড প্রোপাগেশন। এই ধাপে আমরা নিউরাল নেটওয়ার্কের ইনপুট লেয়ারে ইনপুট দেব, নেটওয়ার্ক হিসাবনিকাশ করে আউটপুট লেয়ারে একটি আউটপুট দেবে। অর্থাৎ ফিড ফরওয়ার্ডের ইনপুট হচ্ছে আমাদের ইনপুট ভেষ্টর, আর আউটপুট হচ্ছে আমাদের আউটপুট লেয়ারের আউটপুট ভেষ্টর।

আবার, দ্বিতীয় অ্যালগরিদম হলো ব্যাকপ্রোপাগেশন, যেখানে নিউরাল নেটওয়ার্ক তার আউটপুট লেয়ারে যে আউটপুট দেয় সেটি প্রকৃত আউটপুটের সঙ্গে মিলিয়ে দেখা হয় যে সেটি ঠিক হয়েছে না ভুল। যদি ঠিক হয়ে থাকে, তাহলে আর ওয়েইটের মান আপডেট করার দরকার হয় না। কিন্তু যদি ভুল হয়, তাহলে অবশ্যই আমাদের ওয়েইটের মান আপডেট করতে হবে। পদ্ধতিটি অনেকটাই পারসেপ্ট্রনকে ট্রেইন করার সঙ্গে মিলে যায়। পারসেপ্ট্রনের সঙ্গে পার্থক্য এই যে, পারসেপ্ট্রনে একটিই নিউরন ছিল, আর এই নিউরাল নেটওয়ার্কে অনেকগুলো নিউরন আছে।

তাহলে ব্যাকপ্রোপাগেশনে ইনপুট হচ্ছে আমাদের ফিড ফরওয়ার্ড থেকে পাওয়া আউটপুট ভেষ্টর এবং আমাদের প্রকৃত আউটপুট ভেষ্টর। আর এই ব্যাকপ্রোপাগেশনের আউটপুট হচ্ছে নিউরাল নেটওয়ার্কের আপডেট করা নতুন ওয়েইট ভেষ্টর।

মেশিন লার্নিং অ্যালগরিদম

যদি এমন হতো যে আমাদের নিউরাল নেটওয়ার্কটি আগে থেকেই ট্রেইন করা, একে আমাদের ট্রেনিং দিয়ে নেওয়া লাগবে না, খালি আমরা তাকে ইনপুট ডেটা দেব আর সে আমাদের আউটপুট দেবে, তখন সেই নিউরাল নেটওয়ার্কটির নাম হতো ফিড ফরওয়ার্ড নিউরাল নেটওয়ার্ক। তাতে কোনো ব্যাকপ্রোপাগেশন অ্যালগরিদম থাকত না, যেহেতু ট্রেনিং করে ওয়েইটের মান আপডেট করার মতো কোনো ব্যাপার নেই। শুধু ফিড ফরওয়ার্ড অ্যালগরিদমের মাধ্যমে ইনপুট থেকে আউটপুট বের করা হতো।

নিউরাল নেটওয়ার্কের বিস্তারিত বিবরণ এখানে দিচ্ছি না। আসলে নিউরাল নেটওয়ার্ক নিজেই এখন মেশিন লার্নিংয়ের একটি অনেক বড়ো শাখা হয়ে দাঁড়াচ্ছে। তাই ঠিক করেছি যে, নিউরাল নেটওয়ার্ক নিয়ে সম্পূর্ণ আলাদা একটি লেখা লিখব। সেখানে নিউরাল নেটওয়ার্ক, ডিপ নিউরাল নেটওয়ার্ক, রিকারেন্ট নিউরাল নেটওয়ার্ক, কনভেলিউশনাল নিউরাল নেটওয়ার্ক ইত্যাদি নিয়ে বিশদ বর্ণনা থাকবে। এই বইয়ের জন্য আপাতত নিউরাল নেটওয়ার্ক পরিচিতি এটুকুই।

অধ্যায় ১৩: পারফরম্যান্স (Performance)

আমরা এ পর্যন্ত যতগুলো মেশিন লার্নিং অ্যালগরিদম দেখলাম, সবগুলোই তিনটি ভাগের মধ্যে কোনো একটিতে পড়ে –

- রিগ্রেশন : লিনিয়ার রিগ্রেশন
- ক্লাসিফিকেশন : লজিস্টিক রিগ্রেশন, সাপোর্ট ভেস্টের মেশিন, কে-নিয়ারেস্ট নেইবরস, নাইভ বেইজ ক্লাসিফায়ার, ডিসিশন ট্রি, পারসেপ্ট্রন এবং নিউরাল নেটওয়ার্ক
- ক্লাস্টারিং : কে-মিনস ক্লাস্টারিং

এ ছাড়াও এর বাইরে, ডেটার ডাইমেনশন কমিয়ে আনার জন্য আমরা দেখিয়েছি প্রিসিপাল কম্পোনেন্ট অ্যানালাইসিস।

আমাদের বইতে আমরা এতক্ষণ পর্যন্ত শুধু দেখেছি অ্যালগরিদমগুলো কীভাবে ডেটাসেটের ওপরে ব্যবহার করতে হয়। কিন্তু, অ্যালগরিদমগুলো আদৌ ভালো কাজ করছে কি না, অথবা করলেও কতটুকু ভালো কাজ করছে তা-ও তো বোঝা দরকার। কিংবা ধরি, যদি ডেটাসেটের ওপরে একাধিক ক্লাসিফিকেশন অ্যালগরিদম আমরা প্রয়োগ করি, তাহলে কোন অ্যালগরিদম সবচেয়ে ভালো কাজ করবে সেটিও আমাদের বুঝতে হবে, তাই না?

এজন্য বিভিন্ন অ্যালগরিদমের পারফরম্যান্স নির্ণয় করার জন্য কিছু পারফরম্যান্স মেট্রিক (Performance Metric) আছে, এদের সম্পর্কে আমাদের জানা প্রয়োজন। নিচে এদেরকে একটি টেবিল আকারে দেওয়া হলো :

অ্যালগরিদম টাইপ	Performance Metrics
রিগ্রেশন	R - Squared Value
ক্লাসিফিকেশন	Confusion Matrix, Accuracy, Precision, Recall, Specificity, F1 Measure, ROC Curve
ক্লাস্টারিং	Elbow Method

টেবিল 13.1

এখানে উল্লেখ্য, Elbow Method ঠিক কোনো পারফরম্যান্স মেট্রিক নয়, বরং কে-মিনস ক্লাস্টারিং-এর K অর্থাৎ ক্লাস্টার সংখ্যার অপটিমাল মান বের করার পদ্ধতি।

এটি কে-মিনস ক্লাস্টারিং-এর অধ্যায়ের সঙ্গে জুড়ে না দিয়ে আলাদা করে দেওয়ার পেছনে কারণ হচ্ছে, যাতে একটু আলাদা করে চোখে পড়ে। পদ্ধতিটি একটু ভিন্ন এবং অত্যন্ত গুরুত্বপূর্ণ। তাই এটি এই অধ্যায়ে আলোচনা করা হয়েছে।

তো শুরু করা যাক!

পরিচেদ ১৩.১ : আর-স্কয়ারড ভ্যালু (R-Squared Value)

আমাদের প্রথম কে-মিনস ক্লাস্টারিং হলো আর-স্কয়ারড ভ্যালু (R-Squared Value)। এটি মূলত ব্যবহার করা হয় কোনো রিগ্রেশন মডেল কত ভালোভাবে কাজ করছে, সেটি নির্ণয় করার জন্য। এই R-Squared-এর মান যত বড়ো হবে, বুঝতে হবে যে আমাদের রিগ্রেশন মডেল তত ভালো কাজ করেছে, কিংবা ডেটাতে তত ভালো ফিট করেছে।

এটি বের করার সূত্র হচ্ছে –

$$R^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

এখানে \hat{y}_i , \bar{y} এবং y_i হচ্ছে যথাক্রমে আমাদের মডেলের আন্দাজ করা প্রতিটি মান, সমস্ত প্রকৃত আউটপুট মানের গড়মান এবং প্রতিটি প্রকৃত আউটপুটের মান।

যেমন ধরি, আমাদের রিগ্রেশন মডেল একবার চালানোর পরে আমরা পাই :

অরিজিনাল আউটপুট, y_i	$y_i - \bar{y}$	$(y_i - \bar{y})^2$	মডেলের দেওয়া আউটপুট, \hat{y}	$\hat{y} - \bar{y}$	$(\hat{y} - \bar{y})^2$
2	-2	4	2.8	-1.2	1.44
4	0	0	3.4	-0.6	0.36
5	1	1	4	0	0
4	0	0	4.6	0.6	0.36
5	1	1	5.2	1.2	1.44
$\bar{y} = 4$		$\sum_{i=1}^n (y_i - \bar{y})^2 = 6$	$\sum_{i=1}^n (\hat{y}_i - \bar{y})^2 = 3.6$		

টেবিল 13.1.1

অধ্যায় ১৩ : পারফরম্যান্স (Performance)

সুতরাং, এই ফের্টে, $R^2 = \frac{3.6}{6} = 0.6 = 60\%$ যেটি কিনা কিছুটা মধ্যমগোছের পারফরম্যান্স। এই মান ১-এর যত কাছাকাছি যাবে, আমরা ধরে নেব, আমাদের মডেল তত নিখুঁতভাবে মান আন্দাজ করতে পারছে অর্থাৎ মডেলের অ্যাকুরেসি (Accuracy) তত ভালো হচ্ছে।

পরিচ্ছেদ ১৩.২ : কনফিউশন ম্যাট্রিক্স (Confusion Matrix)

প্রথমেই বলে রাখি, কনফিউশন ম্যাট্রিক্স (Confusion Matrix) নিজে কোনো পারফরম্যান্স মেট্রিক নয়, কিন্তু ক্লাসিফিকেশনের জন্য বাকি যতগুলো পারফরম্যান্স মেট্রিক আছে, সবগুলোই তৈরি হয় এই কনফিউশন ম্যাট্রিক্স থেকে, তাই এটি বোঝা জরুরি।

ধরুন, একজন রোগীর সম্পর্কে বিভিন্ন তথ্য আমরা কোনো মেশিন লার্নিং অ্যালগরিদমকে ইনপুট দিয়েছি এবং সেই অ্যালগরিদমের কাজ হচ্ছে আমাদের দেওয়া তথ্যের ওপরে ভিত্তি করে এটি প্রেডিষ্ট বা নির্ণয় করা যে, ওই রোগীর ক্যানসার আছে কি না। এখন চিন্তা করে দেখুন, এখানে চার ধরনের পরিস্থিতির উভব হতে পারে :

True Positive অ্যালগরিদম বলেছে রোগীর ক্যানসার আছে এবং সত্যিই তাই	False Positive অ্যালগরিদম বলেছে রোগীর ক্যানসার আছে কিন্তু আসলে নেই
False Negative অ্যালগরিদম বলেছে রোগীর ক্যানসার নেই, কিন্তু আসলে আছে	True Negative অ্যালগরিদম বলেছে রোগীর ক্যানসার নেই এবং সত্যিই তাই

টেবিল 13.2.1

এখন, আরেকটু গুচ্ছিয়ে লিখলে ওপরের চার্টটি দাঁড়ায় :

		Actual Data	
		(Positive)	(Negative)
Predicted Data	(Positive)	True Positive	False Positive
	(Negative)	False Negative	True Negative

টেবিল 13.2.2

অধ্যায় ১৩: পারফরম্যান্স (Performance)

সুতরাং, এই ক্ষেত্রে, $R^2 = \frac{3.6}{6} = 0.6 = 60\%$ যেটি কিনা কিছুটা মধ্যমগোছের পারফরম্যান্স।
এই মান ১-এর যত কাছাকাছি যাবে, আমরা ধরে নেব, আমাদের মডেল তত নিখুঁতভাবে মান
আন্দাজ করতে পারছে অর্থাৎ মডেলের অ্যাকুরেসি (Accuracy) তত ভালো হচ্ছে।

পরিচ্ছেদ ১৩.২: কনফিউশন ম্যাট্রিক্স (Confusion Matrix)

প্রথমেই বলে রাখি, কনফিউশন ম্যাট্রিক্স (Confusion Matrix) নিজে কোনো পারফরম্যান্স মেট্রিক নয়, কিন্তু ক্লাসিফিকেশনের জন্য বাকি যতগুলো পারফরম্যান্স মেট্রিক আছে, সবগুলোই তৈরি হয় এই কনফিউশন ম্যাট্রিক্স থেকে, তাই এটি বোঝা জরুরি।

ধরুন, একজন রোগীর সম্পর্কে বিভিন্ন তথ্য আমরা কোনো মেশিন লার্নিং অ্যালগরিদমকে ইনপুট দিয়েছি এবং সেই অ্যালগরিদমের কাজ হচ্ছে আমাদের দেওয়া তথ্যের ওপরে ভিত্তি করে এটি প্রেডিক্ষন বা নির্ণয় করা যে, ওই রোগীর ক্যানসার আছে কি না। এখন চিন্তা করে দেখুন, এখানে চার ধরনের পরিস্থিতির উভব হতে পারে :

True Positive অ্যালগরিদম বলেছে রোগীর ক্যানসার আছে এবং সত্যিই তাই	False Positive অ্যালগরিদম বলেছে রোগীর ক্যানসার আছে কিন্তু আসলে নেই
False Negative অ্যালগরিদম বলেছে রোগীর ক্যানসার নেই, কিন্তু আসলে আছে	True Negative অ্যালগরিদম বলেছে রোগীর ক্যানসার নেই এবং সত্যিই তাই

টেবিল 13.2.1

এখন, আরেকটু গুচ্ছিয়ে লিখলে ওপরের চার্টটি দাঁড়ায় :

		Actual Data	
Predicted Data	(Positive)	(Positive)	(Negative)
	(Positive)	True Positive	False Positive
	(Negative)	False Negative	True Negative

টেবিল 13.2.2

এটিই কনফিউশন ম্যাট্রিক্স। এখন, ভালো করে লক্ষ করবেন, কনফিউশন ম্যাট্রিক্সের ভেতরে আমি নতুন চারটি টার্ম ব্যবহার করেছি। সেগুলোর একটু বর্ণনা দিয়ে নিই :

1. ট্রু পজিটিভ (True Positive) : যখন অ্যালগরিদম কোনো কিছুকে সত্যি বলে প্রেডিক্ট করে এবং সেটি আসলেও সত্যি হয়, তখন তাকে বলা হয় ট্রু পজিটিভ। যেমন, আমাদের এই ক্ষেত্রে, মেশিন যদি বলে রোগীর ক্যানসার আছে এবং যদি আসলেও রোগীর ক্যানসার থাকে, তখন তাকে বলা হবে ট্রু পজিটিভ।
2. ফলস পজিটিভ (False Positive) : যখন অ্যালগরিদম কোনো কিছুকে সত্যি বলে প্রেডিক্ট করে, কিন্তু সেটি আসলে সত্যি নয়, তখন তাকে বলা হয় ফলস পজিটিভ। যেমন, আমাদের এই ক্ষেত্রে, মেশিন যদি বলে রোগীর ক্যানসার আছে, কিন্তু যদি আসলে রোগীর ক্যানসার না থাকে, তখন তাকে বলা হবে ফলস পজিটিভ।
3. ফলস নেগেটিভ (False Negative) : যখন অ্যালগরিদম কোনো কিছুকে মিথ্যা বলে প্রেডিক্ট করে, কিন্তু সেটি আসলে সত্যি হয়, তখন তাকে বলা হয় ফলস নেগেটিভ। যেমন, আমাদের এই ক্ষেত্রে, মেশিন যদি বলে রোগীর ক্যানসার নেই, কিন্তু যদি আসলে রোগীর ক্যানসার থাকে, তখন তাকে বলা হবে ফলস নেগেটিভ।
4. ট্রু নেগেটিভ (True Negative) : যখন অ্যালগরিদম কোনো কিছুকে মিথ্যা বলে প্রেডিক্ট করে এবং সেটি আসলেই মিথ্যা হয়, তখন তাকে বলা হয় ট্রু নেগেটিভ। যেমন, আমাদের এই ক্ষেত্রে, মেশিন যদি বলে রোগীর ক্যানসার নেই এবং যদি আসলেও রোগীর ক্যানসার না থাকে, তখন তাকে বলা হবে ট্রু নেগেটিভ।

এখান থেকে আমরা পরিষ্কার বুঝতে পারছি যে ফলস পজিটিভ এবং ফলস নেগেটিভ আমাদের মোটেও কাম্য নয়। এখন পরিস্থিতি অর্থাৎ প্রবলেমের ধরনের ওপরে ভিত্তি করে আমাদের ঠিক করতে হবে যে আমরা ফলস পজিটিভের হার কমাব, নাকি ফলস নেগেটিভের, নাকি দুটোই কমানোর পদক্ষেপ নেব।

যেমন, যদি ক্যানসারের রোগীর উদাহরণ থেকে বলি, ফলস পজিটিভ মানে কোনো রোগীর ক্যানসার নেই, তাকে ভুল করে বলা হয়েছে তার ক্যানসার আছে। এর ফলে কী হবে, তার ক্যানসার না থাকা সত্ত্বেও হয়তো তার ক্যানসারের চিকিৎসা করা হবে। এটি অবশ্যই খারাপ এবং ক্ষতিকর।

কিন্তু চিন্তা করে দেখুন, যদি আমরা ফলস নেগেটিভ পাই, তাহলে কী হবে? কোনো রোগীর ক্যানসার থাকা সত্ত্বেও মেশিন রিপোর্ট দেবে তার ক্যানসার নেই। সেই রোগী হয়তো সেই ভুল রিপোর্ট নিয়ে খুশিমনে বাসায় চলে যাবে এবং কিছুদিনের মধ্যেই ক্যানসারে ভুগে মারা যাবে।

সুতরাং, দুটোর মধ্যে তুলনা করে আমরা দেখতে পাই যে, এই ক্ষেত্রে, দু-একটা ফলস পজিটিভ চলে এলে যতটুকু ক্ষতি হবে, তার চেয়ে বরং একটিও যদি ফলস নেগেটিভ থাকে, তাহলে তাতে

ব্যবহার করতে পারি। অর্থাৎ, ধরা যাক, আমাদের কাছে থাকা 100টি ছবির মধ্যে থেকে আমরা বের করতে চাইছি কোনটি আপেল আর কোনটি কমলার ছবি। এখন যদি, এই ডেটাসেটে মোটামুটি 50% থাকে আপেলের ছবি আর বাকি 50% থাকে কমলার ছবি – এরকম ক্ষেত্রে অ্যাকুরেসি ব্যবহার করা যায়।

কিন্তু যদি, আমাদের ডেটাসেটের টার্গেট ভ্যারিয়েবলগুলো ভালোভাবে ব্যালান্সড না থাকে, তখন অ্যাকুরেসি ব্যবহার করা ঠিক হবে না। ধরা যাক, আমাদের কাছে 100 জন সন্তান্য ক্যানসার রোগীর ডেটা আছে, এর মধ্যে 95 জনের (95%) ক্যানসার নেই, আর বাকি 5 জনের (5%) সত্যি সত্যিই ক্যানসার ধরা পড়েছে।

এখন, ধরা যাক, আমরা যেই অ্যালগরিদম তৈরি করেছি সেটি খুবই অকার্যকর, একেবারেই ক্যানসার রোগী ঠিকমতো চিনতে পারে না। সে ক্ষেত্রে সে এই 100 জন রোগীর ক্ষেত্রে আউটপুট দেবে যে 100 জনের কেউই ক্যানসারে আক্রান্ত নয়।

এখন আমরা যদি এ ক্ষেত্রে শুধু অ্যাকুরেসি মাপি, তাহলে আমরা হয়তো বলে বসব যে আমাদের মডেল খুব ভালো, একেবারে 95% অ্যাকুরেসি অর্জন করেছে, কিন্তু আসলেই কি তাই? আমাদের 5 জন ক্যানসার রোগী ছিল, মডেল একজনকেও চিনতে পারেনি। তার মানে, ক্যানসার নির্ণয়ে সে সম্পূর্ণ ব্যর্থ হয়েছে, সুতরাং এ ক্ষেত্রে শুধু অ্যাকুরেসি দিয়ে মডেলের ভালো/মন্দ বিচার করলে হবে না।

মোটামুটি এই-ই ছিল অ্যাকুরেসির ধারণা। আশা করি সবাই বুঝতে পেরেছেন।

পরিচ্ছেদ ১৩.৪ : প্রিসিশন (Precision) ও রিকল (Recall)

আমাদের পরিচ্ছেদ ১৩.২-এ ব্যবহৃত কনফিউসন ম্যাট্রিক্সটি ছিল এরকম :

		Actual Data	
		(Positive)	(Negative)
Predicted Data	(Positive)	True Positive	False Positive
	(Negative)	False Negative	True Negative

টেবিল 13.4.2

প্রিসিশন (Precision)-এর ধারণাটি আমরা এখান থেকেই নেব। প্রিসিশন মানে হচ্ছে, আমাদের ক্যানসারের উদাহরণের সঙ্গে তাল মিলিয়ে যদি বলতে চাই – অ্যালগরিদম যতজন রোগীকে

অধ্যায় ১৩ : পারফরম্যান্স (Performance)

'ক্যানসার আছে' বলে ঘোষণা দিয়েছে এবং তাদের সবার মধ্যে, যাদের আসলেই ক্যানসার আছে, এদের একটি অনুপাত।

অর্থাৎ,

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

উদাহরণস্বরূপ, ধরা যাক, আমাদের 100 জন সন্তানের মধ্যে মাত্র 5 জনের আসলেই ক্যানসার আছে, বাকি 95 জনের নেই। এখন ধরা যাক, আমাদের অ্যালগরিদম খুবই অকার্যকর এবং সে সবাইকে ক্যানসার রোগী হিসেবে সন্দেহ করছে। সুতরাং, অ্যালগরিদম রিপোর্ট দেবে যে, 100 জনের সবাইই ক্যানসার রোগী। অর্থাৎ,

$$True\ Positive + False\ Positive = 100$$

কিন্তু, বাস্তবে রোগী হচ্ছে 5 জন, অর্থাৎ,

$$True\ Positive = 5$$

সুতরাং

$$Precision = \frac{5}{100} = 0.05 = 5\%$$

এখন আসি রিকল (Recall)-এর ক্ষেত্রে কী হবে সেটাতে। রিকল হচ্ছে মূলত অ্যালগরিদম সত্যিকারের ক্যানসার রোগীদের মধ্যে কতজনকে ঠিক ঠিক প্রেডিক্ট করতে পেরেছে এবং সত্য সত্যিই কতজনের ক্যানসার আছে (অ্যালগরিদম সেটাকে ঠিকমতো প্রেডিক্ট করে থাকুক, কিংবা না-ই থাকুক) তার একটি অনুপাত।

অর্থাৎ,

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

এখানে ফলস নেগেটিভও যোগ করার কারণ হচ্ছে, ফলস নেগেটিভদের ক্ষেত্রে সত্য সত্য এখানে ফলস নেগেটিভও যোগ করার কারণ হচ্ছে, ফলস নেগেটিভদের ক্ষেত্রে সত্য সত্য সত্য সত্য।

ক্যানসার থাকা সত্ত্বেও অ্যালগরিদম তাদের ক্ষেত্রে বলবে যে ক্যানসার নেই।
যদি, 100 জনের ভেতরে 10 জনের সত্যিই ক্যানসার থাকে এবং তাদের মধ্যে 5 জনকে অ্যালগরিদম ঠিক ঠিক বের করে ফেলতে পারে, তাহলে –

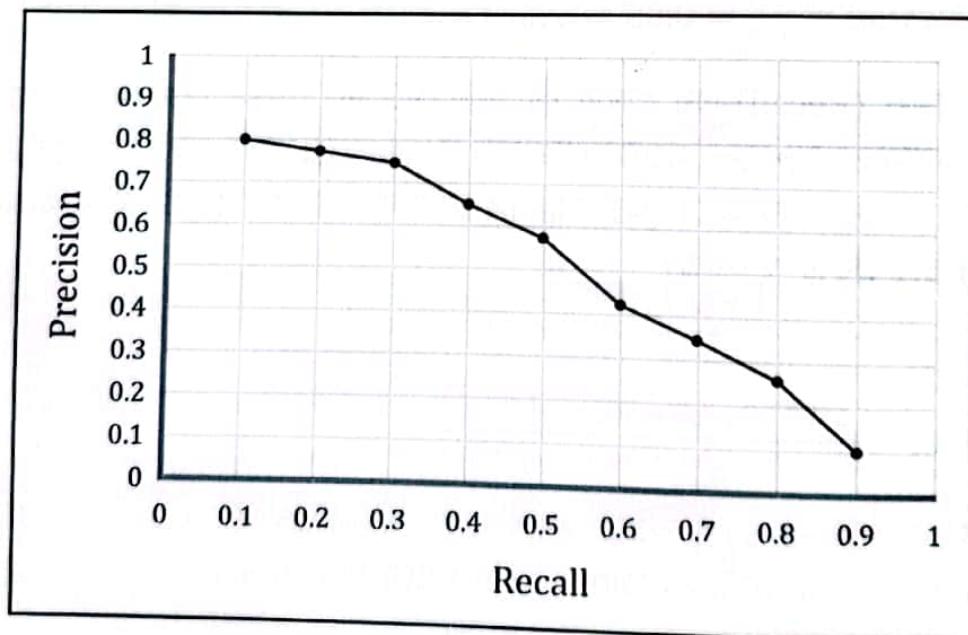
$$Recall = \frac{5}{10} = \frac{1}{2} = 50\%$$

মেশিন লার্নিং অ্যালগরিদম

এতক্ষণের আলোচনা থেকে এটি কিন্তু পরিষ্কার বোঝা যাচ্ছে যে, প্রিসিশন কাজ করবে ফলস পজিটিভ নিয়ে, অর্থাৎ বোঝাবে যে আমরা কতজন সত্যিকারের ক্যানসার রোগীকে নির্ণয় করতে পেরেছি। অন্যদিকে রিকল কাজ করবে ফলস নেগেটিভ নিয়ে, অর্থাৎ আমরা কতজন সত্যিকারের ক্যানসার রোগীকে ঠিকমতো বের করতে পারিনি, সেটি বোঝাবে।

তাই, আমরা যদি চাই আমাদের সিস্টেমের ফলস নেগেটিভ কমাতে, তখন আমাদের চেষ্টা করতে হবে যে আমাদের রিকলের মান যেন 100%-এর কাছাকাছি নিয়ে যাওয়া যায়। যত কাছাকাছি নিতে পারব, আমাদের ফলস নেগেটিভ তত কমে আসতে থাকবে।

এখানে একটু খেয়াল রাখতে হবে, রিকল বাড়ানোর জন্য, অর্থাৎ, আমাদের যাতে কোনো ক্যানসার রোগী বাদ না পড়ে, সেটি নিশ্চিত করার জন্য আমরা কী করব? যত বেশি সন্তুষ মানুষকে ক্যানসার রোগী হিসেবে প্রেডিস্ট করানোর চেষ্টা করব, তাই না? যত বেশি মানুষ আমরা ক্যানসার রোগী হিসেবে নেব, ততই আমাদের আসল ক্যানসার রোগীদের বাদ পড়ার সন্তাবনা কমে আসবে, ঠিক? কিন্তু সেই সঙ্গে দেখুন, ক্যানসারে আক্রান্ত নয়, এমন রোগীও ক্যানসার রোগী হিসেবে চিহ্নিত হওয়ার সন্তাবনা এবং সংখ্যা বাড়তে থাকবে, অর্থাৎ ফলস পজিটিভ বাড়তে থাকবে। তাই রিকল বাড়ানোর সঙ্গে সঙ্গে এটিও লক্ষ রাখতে হবে, যাতে প্রিসিশন খুব বেশি কমে না যায়, অর্থাৎ অনেক বেশি ফলস পজিটিভ বেড়ে না যায়।



গ্রাফ 13.4.1 : প্রিসিশন-রিকল গ্রাফ

একইভাবে, যদি আমরা চাই ফলস পজিটিভ কমাতে, আমাদের তখন চেষ্টা করতে হবে প্রিসিশনের মান 100%-এর যত কাছাকাছি নিয়ে যাওয়া যায়। এখন ফলস পজিটিভ কমানোর জন্য আমরা কী করব? আমাদের মোট ক্যানসার রোগী হিসেবে চিহ্নিত রোগীর সংখ্যা কমানোর চেষ্টা করব, যাতে আসল ক্যানসার রোগী বাদে অন্য কেউ না থাকে, তাই না? কিন্তু অনেক সময় এই ক্যানসার

অধ্যায় ১৩ : পারফরম্যান্স (Performance)

রোগী হিসেবে চিহ্নিত রোগীর সংখ্যা কমাতে গিয়ে আমরা আসল ক্যানসার রোগীকেও বাদ দিয়ে দিতে পারি, যেটি কিনা ফলস নেগেটিভ বাড়িয়ে দেবে। তাই, প্রিসিশন বাড়ানোর সঙ্গে সঙ্গে এটিও নিশ্চিত করতে হবে, যাতে রিকল খুব বেশি কমে না যায়। গ্রাফ 13.4.1-এ ব্যাপারটি আরেকটু সহজ করে বোঝানোর চেষ্টা করা হলো :

গ্রাফ 13.4.1 থেকে ভালোমতো লক্ষ করলে দেখবেন, রিকলের মান যখন খুবই কম, প্রিসিশনের মান তখন অনেক বেশি। আন্তে আন্তে রিকলের মান বাড়ানোর সঙ্গে সঙ্গে আমাদের প্রিসিশনের মান কমেছে। এটিকে বলে প্রিসিশন-রিকল ট্রেডঅফ (Precision-Recall Tradeoff)। তাই, আমাদের সিস্টেমের চাহিদা অনুসারে আমাদের ঠিক করে নিতে হবে যে আমরা প্রিসিশনকে ম্যাক্সিমাইজ করব নাকি রিকলকে।

এই ছিল প্রিসিশন এবং রিকলসংক্রান্ত আলোচনা।

পরিচ্ছেদ ১৩.৫ : এফ-ওয়ান মেজার (F1 Measure)

আমরা এতক্ষণ দেখলাম, কী করে প্রিসিশন এবং রিকল নির্ণয় করতে হয় এবং কখন কোনটি কীভাবে বাড়াতে কিংবা কমাতে হয়। কথা হচ্ছে, প্রিসিশন ও রিকল দুটি সম্পূর্ণ ভিন্ন ভিন্ন পারফরম্যান্স মেট্রিক। এখন একটি অ্যালগরিদমকে যাচাই করার জন্য যদি এই দুটি ভিন্ন ভিন্ন মানের পরিবর্তে দুটি মিলিয়ে একটি মান ব্যবহার করা যায়, সেটি আরো ভালো হয় না?

ঠিক এই ধারণার থেকেই এফ-ওয়ান মেজার (F1 Measure)-এর জন্ম। এফ-ওয়ান মেজারকে সহজ করে বলা যায় যে, এটি হচ্ছে প্রিসিশন ও রিকল এর মানের 'এক বিশেষ ধরনের' গড়। এখন কথা হচ্ছে, বিশেষ ধরনের গড় আবার কী? গড় তো গড়ই। দুটি মান যোগ করব, দুই দিয়ে ভাগ করব, এই তো হলো গড়! এর বাইরেও আবার কিছু আছে নাকি?

তার উত্তর দেওয়ার আগে চলুন একটু দেখে নিই প্রিসিশন ও রিকলের সাধারণ গড় মান নিলে কোন পরিস্থিতির উদ্দেক হয়।

আমরা যদি সাধারণ গাণিতিক গড় নিই, তবে –

$$F1\ Measure = \frac{Precision + Recall}{2}$$

এটি কিছু কিছু ক্ষেত্রে ভালো কাজ করলেও কিছু কিছু ক্ষেত্রে খুবই ক্রটিপ্রবণ ফলাফল দেবে। যেমন, ধরা যাক, আমাদের 100 জন রোগীর মধ্যে 3 জনের ক্যানসার আছে, বাকি 97 জনের রোগী হিসেবে প্রেডিষ্ট করেছে। তাহলে –

		Actual Data	
Predicted Data		(Cancer)	(No Cancer)
	(Cancer)	3	97
	(No Cancer)	0	0

টেবিল 13.5.1

এখন, তাহলে এই টেবিল থেকে আমরা যদি প্রিসিশন ও রিকল বের করে ফেলি তাহলে দাঁড়ায় এরকম –

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} = \frac{3}{100} = 3\%$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = \frac{3}{3} = 100\%$$

সুতরাং এখান থেকে আমাদের এফ-ওয়ান মেজার হয়,

$$F1 \text{ Measure} = \frac{3 + 100}{2} = 51.5\%$$

এখন দেখুন, আমাদের মডেলটি কিন্তু খুবই অকার্যকর ছিল, সেসব রোগীকেই ক্যানসার রোগী হিসেবে প্রে�িক্ট করেছে। কিন্তু তা সত্ত্বেও আমাদের এফ-ওয়ান মেজার আসছে 51.5% যেটি কিনা বেশ মাঝামাঝি পর্যায়ের একটি মান, তাই না? অথচ এত বাজে মডেলের ক্ষেত্রে অনেক কম থাকার কথা। আর তাই, আমরা শুধু গাণিতিক গড় ব্যবহার করে এফ-ওয়ান মেজারের মান হিসাব করলে সব সময় সঠিক তথ্য পাব না। সুতরাং, আমাদের সেই ‘বিশেষ গড়’-এ ফিরে যেতে হবে।

আমাদের বিশেষ এই গড়মান্টির একটি গালভরা নাম আছে। একে বলা হয় হারমনিক গড় বা হারমনিক মিন (Harmonic Mean)।

হারমনিক গড়ের বৈশিষ্ট্য হলো যদি, আমরা যদি X ও Y সাধারণ গড় (arithmetic mean) হিসাব করি, তাহলে দেখুন এটি X ও Y -এর ঠিক মাঝ বরাবর থাকবে। কিন্তু, হারমনিক গড়ের ক্ষেত্রে সেটি একটু আলাদা হবে। যদি X ও Y -এর মান মোটামুটি সমান হয়, তবে সে ক্ষেত্রে হারমনিক গড়ের মান দুজনের সাধারণ গড়, অর্থাৎ মাঝ বরাবর থাকবে। কিন্তু যদি, এমন হয় যে X ও Y -এর মধ্যে যে-কোনো একটি বড়ো এবং অপরটি ছোটো, তখন হারমনিক গড়ের মান X ও Y -এর মধ্যে যেটি ছোটো সেটির কাছাকাছি থাকবে। অর্থাৎ, যদি X ছোটো হয়, তাহলে X -এর কাছাকাছি থাকবে, কিংবা Y ছোটো হলে Y -এর কাছাকাছি থাকবে।

এই হারমনিক গড়ের সূত্র হচ্ছে,

অধ্যায় ১৩ : পারফরম্যান্স (Performance)

$$\text{Harmonic Mean} = \frac{2XY}{X+Y}$$

অর্থাৎ যদি আমরা প্রিসিশন ও রিকল-এর হারমনিক গড় নিই তাহলে –

$$\text{Harmonic Mean} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

তাহলা এখন, আমাদের সেই ক্যানসারের উদাহরণে প্রিসিশন ও রিকল-এর হারমনিক গড় নিলে গাই –

$$\text{Harmonic Mean} = \frac{2 * 3 * 100}{3 + 100} = \frac{600}{103} = 5.825\%$$

বলুকি মনে হচ্ছে না, যে এই স্কোরটি আমাদের ওই মডেলের জন্য উপযুক্ত হয়েছে? ঠিক তাই।
সব মডেলের পারফরম্যান্স খারাপ, প্রতিটি মডেলকে এভাবে স্কোর করিয়ে দিয়ে উপযুক্ত শাস্তি
য় এই হারমনিক গড়।

গাই, এফ-ওয়ান মেজারের সময় আমরা অবশ্যই সাধারণ গড়ের পরিবর্তে হারমনিক গড় ব্যবহার
করব।

পরিচ্ছন্দ ১৩.৬ : স্পেসিফিসিটি (Specificity)

আবারও, আমাদের পূর্বে ব্যবহৃত টেবিল থেকে –

		Actual Data	
Predicted Data	(Positive)		(Negative)
	(Positive)	True Positive	False Positive
	(Negative)	False Negative	True Negative

টেবিল 13.6.1

স্পেসিফিসিটি (Specificity) হচ্ছে যতজনের ক্যানসার ছিল না বলে মেশিন প্রেডিক্ট করেছে এবং
তাদের মধ্যে আসলেই যারা ক্যানসার রোগী নয়, তাদের একটি অনুপাত। এটি দেখা যাচ্ছে
রিকলের ঠিক বিপরীত।

অর্থাৎ,

$$\text{Specificity} = \frac{\text{True Negative}}{\text{True Negative} + \text{False Positive}}$$

মেশিন লার্নিং অ্যালগরিদম

এখানে ফলস পজিটিভও যোগ করার কারণ হচ্ছে, যারা ফলস পজিটিভ তাদের কিন্তু আসলে ক্যানসার নেই, কিন্তু ভুল ডায়াগনোসিস করে মেশিন বলেছে তাদের ক্যানসার আছে।

ধরা যাক, আগের মতোই, 100 জনের ভেতরে 5 জনের ক্যানসার আছে, বাকি 95 জনের নেই। এখন মেশিন বলল ওই 5 জনসহ মোট 10 জনের ক্যানসার আছে, অর্থাৎ 90 জনের ক্যানসার নেই। অতএব,

$$\text{True Negative} = 5; \quad \text{False Positive} = 5$$

তাহলে,

$$\text{Specificity} = \frac{5}{10} = 50\%$$

এই হচ্ছে স্পেসিফিসিটি। আমাদের লক্ষ্য থাকবে স্পেসিফিসিটির মান যতদূর সন্তুষ্ট বাঢ়ানো।

পরিচ্ছেদ ১৩.৭ : আরওসি কার্ভ (ROC Curve)

এতক্ষণ আমরা যা যা পড়েছি, মোটামুটি সবকিছুর জ্ঞান আমাদের প্রয়োজন হবে আরওসি কার্ভ (ROC Curve) বোঝার জন্য। আরওসি কার্ভের পুরো নাম – রিসিভার অপারেটিং ক্যারেকটারিস্টিক কার্ভ (Receiver Operating Characteristic Curve)। নাম দেখে হয়তো মনে হতে পারে যে বেশ জবরিজৎ একটি পারফরম্যান্স মেট্রিক এটি এবং আসলেই তা-ই।

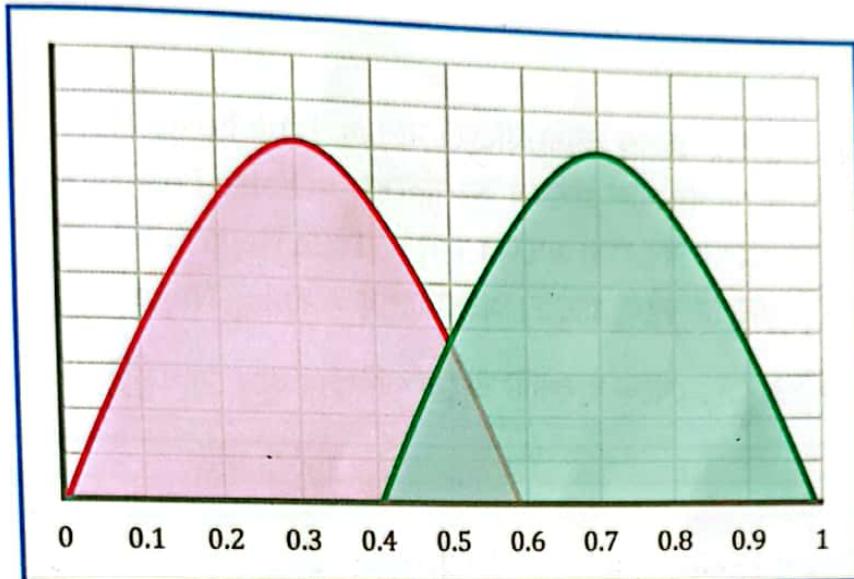
আরওসি কার্ভ দিয়ে আমরা মূলত যেটি বুঝি, সেটি হচ্ছে একটি মডেল কত ভালোভাবে দুটি ভিন্ন ভিন্ন ক্লাসের মধ্যে পার্থক্য করতে পারে। একটি খুব ভালো মডেল খুব ভালোভাবে দুটি ক্লাসের মধ্যে পার্থক্য করতে পারবে, কিন্তু একটি খারাপ মডেল কখনোই তা পারবে না।

আরওসি কার্ভের ধারণায় যাওয়ার আগে, আমরা আমাদের TP, TN, FP ও FN-এর ধারণাটি একটু ভিন্নভাবে দেখি এবার।

ধরা যাক, মেশিন আবারও আগের মতোই ক্যানসারের রোগী প্রেডিষ্ট করছে। এবার আমরা তার অনুমানকে একটি সন্তাব্যতার মান দিয়ে চিহ্নিত করব। গ্রাফ 13.7.1 দেখি। এখানে, লাল রঙের কার্ভ থেকে দেখতে পাচ্ছি, যাদের সন্তাব্যতার মান 0 থেকে 0.6-এর মধ্যে আছে তাদের সত্যি সত্যিই ক্যানসার নেই। আর সবুজ রঙের কার্ভ থেকে দেখতে পাচ্ছি, যাদের সন্তাব্যতার মান 0.4 থেকে 1-এর মধ্যে আছে তাদের সত্যি সত্যিই ক্যানসার আছে। সন্তাব্যতার মানটি কোনোভাবে হিসাবনিকাশ করে বের করা হয়েছে বলে ধরে নেওয়া যাক।

অধ্যায় ১৩: পারফরম্যান্স (Performance)

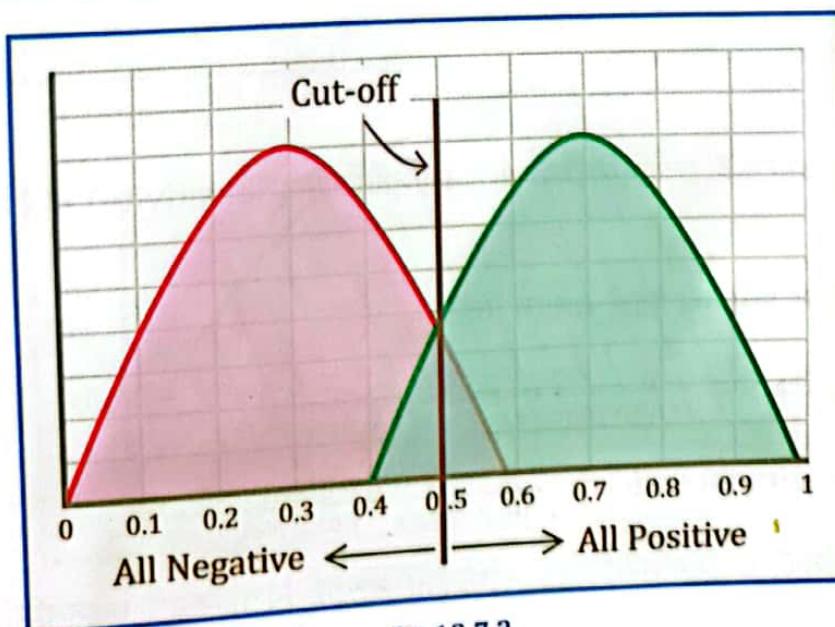
এখন কথা হচ্ছে, ভালো করে লক্ষ করবেন, এখানে কার্ড দুটির কিছু অংশ ওভারল্যাপ (Overlap) করবে। সন্তাব্যতা যদি ০.৪ থেকে ০.৬-এর মধ্যে কোনো মান হয়, তখন কিন্তু পজিটিভ কিংবা নেগেটিভ দুটির যে-কোনোটিই আউটপুট হতে পারে। কিন্তু, তা তো হতে দেওয়া যায় না! কেউ তে একই সঙ্গে পজিটিভ ও নেগেটিভ দুটিই হতে পারে না।



গ্রাফ 13.7.1

তাই, আমরা এখন যেটি করব, সেটি হচ্ছে একটি থ্রেসহোল্ড মান ঠিক করব এবং এটি মেশিনকে শিখিয়ে দেব। থ্রেসহোল্ড মানের নিচের সব সন্তাব্যতার জন্য মেশিন বলবে ক্যানসার নেই; আর থ্রেসহোল্ড মানের ওপরের সব মানের জন্য মেশিন বলবে ক্যানসার আছে।

এখন, আমরা যদি থ্রেসহোল্ড মান ধরি ০.৫, তাহলে –



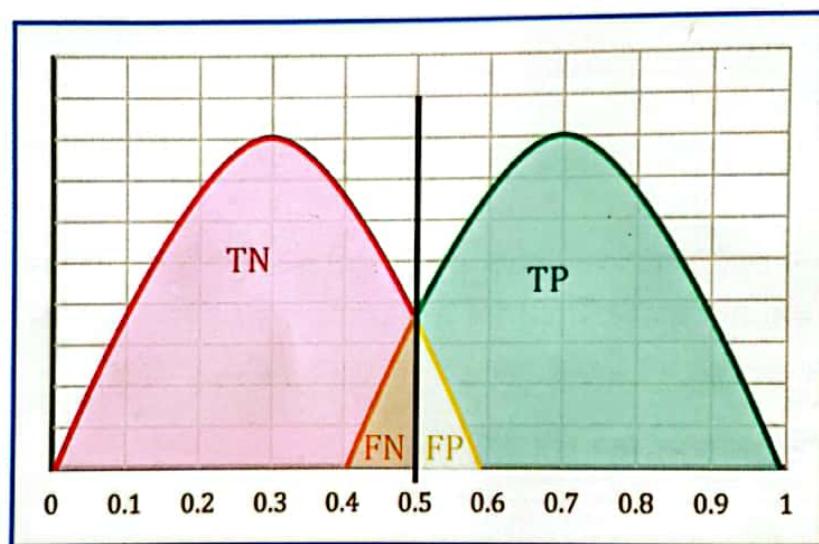
গ্রাফ 13.7.2

এখন ভালোমতো লক্ষ করুন, 0.5-এর দুই পাশেই এমন কিছু মান আছে, যেগুলোর সেখানে থাকার কথা নয়। 0.5-এর ডান পাশে শুধুই সবুজ গ্রাফ থাকার কথা, লাল গ্রাফের কোনো অংশ থাকার কথা নয়। একইভাবে, 0.5-এর বাঁ পাশে শুধুই লাল গ্রাফ থাকার কথা, সবুজ গ্রাফের কোনো অংশ থাকার কথা নয়।

এখন তাহলে বুবাতেই পারছেন,

- 0.5-এর ডান পাশের সব সবুজ গ্রাফের মান = True Positive
- 0.5-এর ডান পাশের লাল গ্রাফের অংশবিশেষ = False Positive
- 0.5-এর বাঁ পাশের সব লাল গ্রাফের মান = True Negative
- 0.5-এর বাঁ পাশের সবুজ গ্রাফের অংশবিশেষ = False Negative

নিচের গ্রাফ (গ্রাফ 13.7.3) থেকে ধারণাটি আরো পরিষ্কার হবে বোধকরি :



গ্রাফ 13.7.3

এখন আমরা ইতিমধ্যেই স্পেসিফিসিটি ও সেনসিটিভিটি (Sensitivity) বা রিকল বিষয় দৃঢ় জানি।

আপনাদের সুবিধার জন্য আবারও এখানে দিচ্ছি –

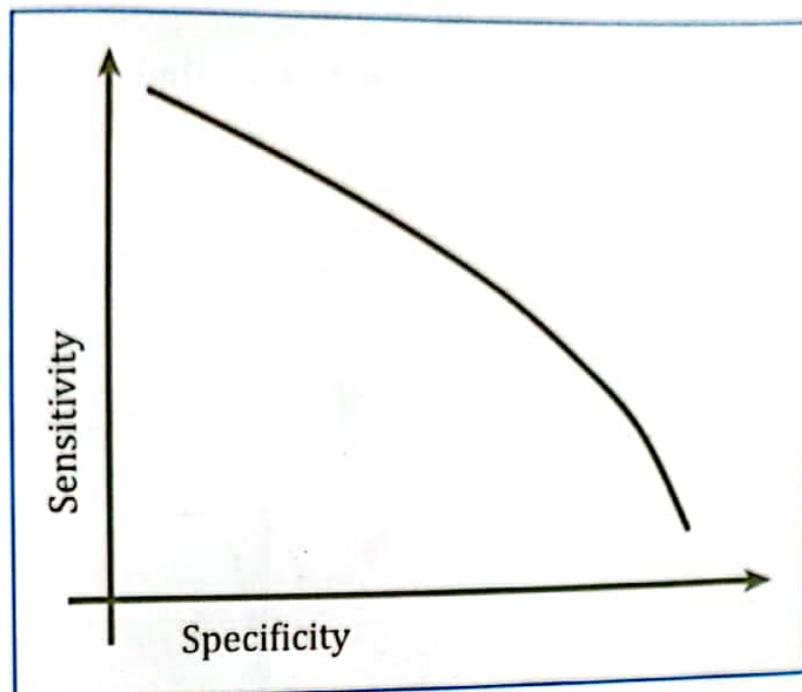
$$\text{Specificity} = \frac{\text{True Negative}}{\text{True Negative} + \text{False Positive}}$$

$$\text{Sensitivity (Recall)} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

এখন স্পেসিফিসিটি ও সেনসিটিভিটির মধ্যেকার সম্পর্ক বিপরীতমুখী। সেনসিটিভিটি বাড়লে স্পেসিফিসিটি কমে, আবার সেনসিটিভিটি কমলে স্পেসিফিসিটি বাড়ে।

অধ্যায় ১৩ : পারফরম্যান্স (Performance)

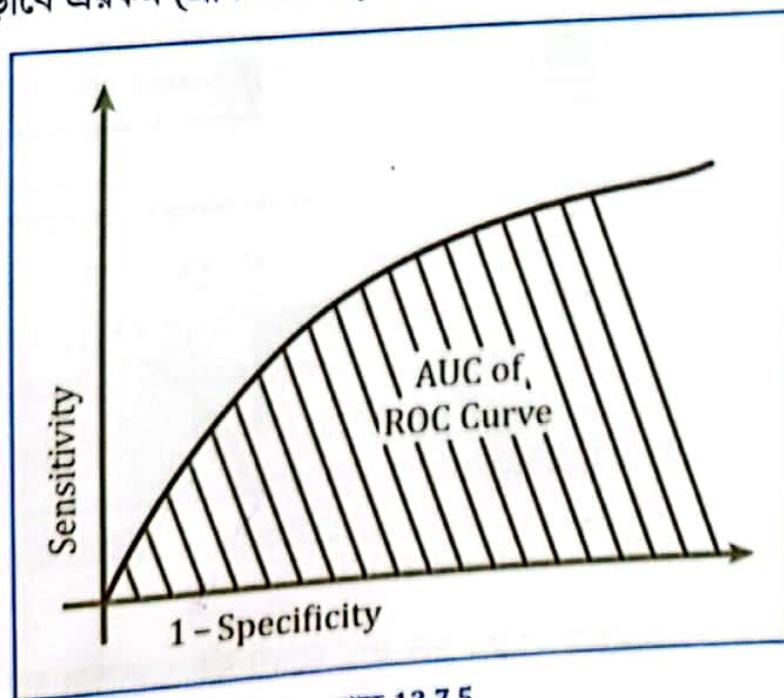
নিচে গ্রাফ (গ্রাফ 13.7.4) থেকে হয়তো বিষয়টি আরেকটু পরিষ্কার হবে :



গ্রাফ 13.7.4

এখন, ROC Curve-এর জন্য আমরা শুধু Specificity-এর মানের বদলে ($1 - Specificity$)-এর মান Sensitivity-এর বিপরীতে প্লট করব।

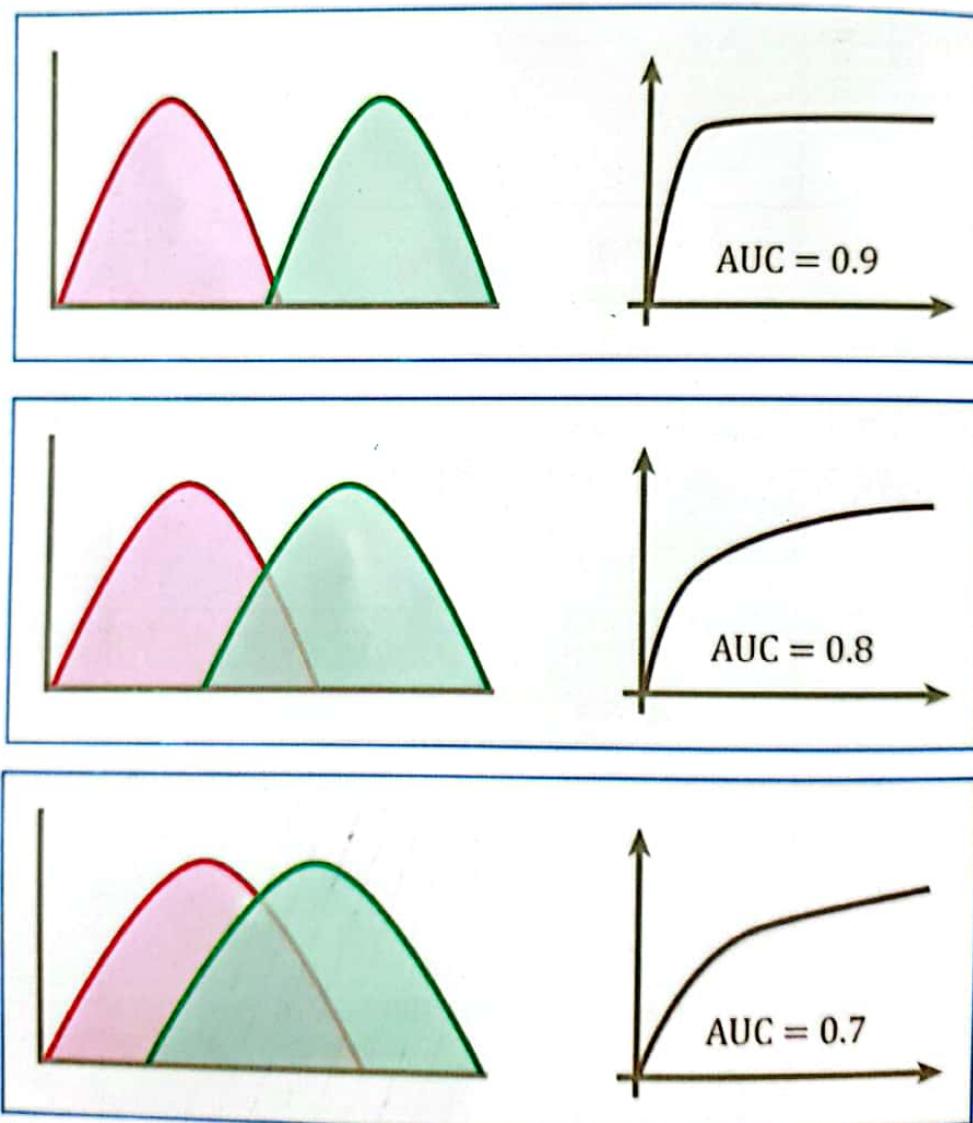
তখন গ্রাফটি দাঁড়াবে এরকম (গ্রাফ 13.7.5) :



গ্রাফ 13.7.5

এই আরওসি কার্ড গ্রাফে তাহলে দেখা যাচ্ছে সেনসিটিভিটি যত বাঢ়বে, ($1 - Specificity$)-এর মানও তত বাঢ়বে। আরেকটি টার্ম আমি এখানে ব্যবহার করেছি, দেখবেন টার্মটি হচ্ছে AUC। এখানে, AUC হচ্ছে এরিয়া আন্ডার দ্য আরওসি কার্ড (Area Under the ROC Curve) বা সংক্ষেপে এরিয়া আন্ডার কার্ড (Area Under Curve - AUC)।

আমাদের, এই আরওসি কার্ড গ্রাফে AUC অর্থাৎ কার্ডের নিচের ক্ষেত্র যত বড়ো হবে, আমাদের মডেল তত ভালো কাজ করছে বলে ধরা হবে। আর ক্ষেত্র যদি কম হয়, তাহলে ধরে নিতে হবে যে আমাদের মডেল দুটি ক্লাসের মধ্যে পার্থক্য করার ক্ষেত্রে খুব একটা ভালো পারফরম করছে না।

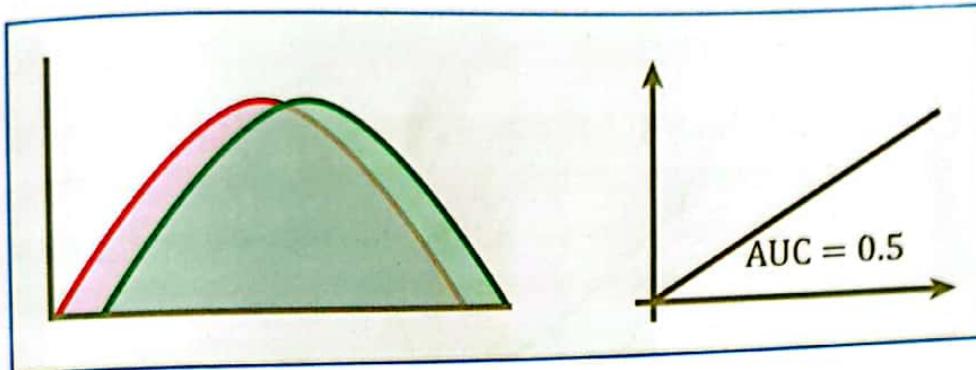


গ্রাফ 13.7.6

ওপরের গ্রাফ (গ্রাফ 13.7.6) থেকে দেখুন, আমাদের মডেল যত খারাপ পারফরম করছে, অর্থাৎ ওভারল্যাপ যত বাঢ়ছে, আমাদের আরওসি কার্ডের AUC ততই কমছে।

অধ্যায় ১৩ : পারফরম্যান্স (Performance)

আবে দেখুন, মডেল যদি প্রায় পুরোপুরিই খারাপ পারফরম করে, অর্থাৎ প্রায় পুরোপুরিই ওভা প্যাপ করে, যদি দুটি ক্লাসের ভেতরে একেবারেই আলাদা না করতে পারে, তখন গ্রাফ হয় এবং (গ্রাফ 13.7.7) :



গ্রাফ 13.7.7

এখন সবশেষে আসি, কেন আমরা ($1 - Specificity$) ব্যবহার করছি, তার ব্যাখ্যায়।

আমরা জানি,

$$Specificity = \frac{TN}{TN + FP}$$

$$1 - Specificity = 1 - \frac{TN}{TN + FP} = \frac{TN + FP - TN}{TN + FP} = \frac{FP}{TN + FP}$$

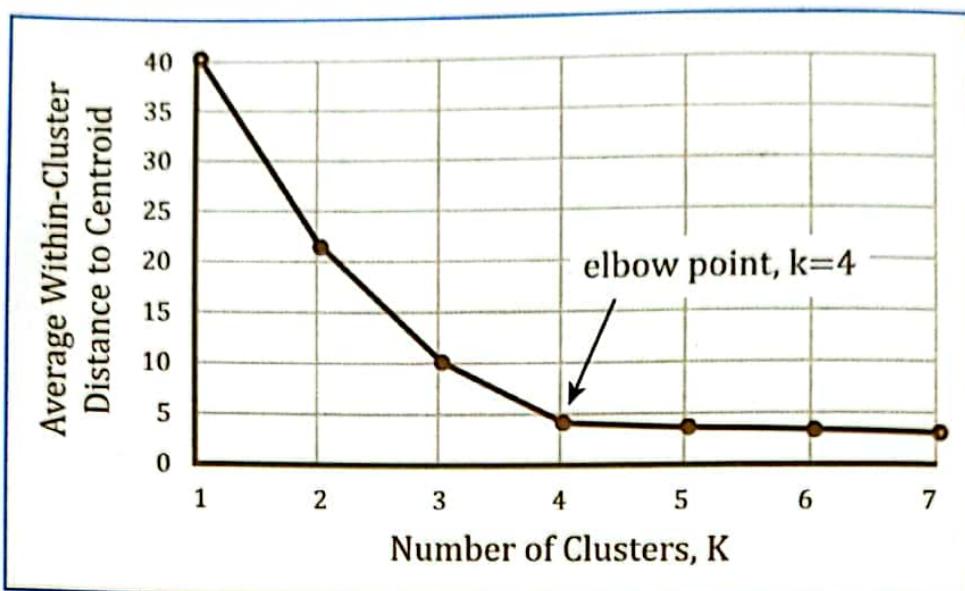
এখন থেকে বোঝা যাচ্ছে, স্পেসিফিটি আমাদের দিত ট্রু নেগেটিভের হার, কিন্তু ($1 - Specificity$) আমাদের দেবে ফলস পজিটিভের হার। সুতরাং, সেনসিটিভিটিকে ট্রু পজিটিভের হার এবং ($1 - Specificity$)-কে ফলস পজিটিভের হার বলা চলে।

এখন তাহলে, আমরা শুধু পজিটিভ নিয়ে কাজ করছি, কোনো ধরনের নেগেটিভ নিয়ে কাজ করছি না। গ্রাফ 13.7.3 থেকে, আমরা যদি আমাদের থ্রেসহোল্ডের মান বাঢ়াই, আমরা তাহলে ট্রু পজিটিভ ও ফলস পজিটিভ দুটিই কমিয়ে নিয়ে আসব। আবার যদি থ্রেসহোল্ডের মান কমাই, পজিটিভ ও ফলস পজিটিভ দুটিই বাঢ়িয়ে ফেলব। অর্থাৎ, আমরা বলতে পারি তাহলে আমরা ট্রু পজিটিভ ও ফলস পজিটিভ দুটিই বাঢ়িয়ে ফেলব। অর্থাৎ, আমরা বলতে পারি যে, আমরা এখন শুধু থ্রেসহোল্ডের ডান পাশের দুটি মান (ট্রু পজিটিভ ও ফলস পজিটিভ) নিয়েই রকমের সুবিধা হয় দেখেই ($1 - Specificity$) ব্যবহার করা হয়ে থাকে।

পরিচেদ ১৩.৮ : এলবো মেথড (Elbow Method)

এলবো মেথড (Elbow Method) ব্যবহার করা হয় কে-মিনস ক্লাস্টারিংয়ের ক্ষেত্রে।

আগেই বলা হয়েছে, এটি আসলে ঠিক কোনো পারফরম্যান্স মেট্রিক নয়। এটি হলো মূলত কে-মিনস ক্লাস্টারিংয়ের ক্ষেত্রে K-এর অপটিমাল মান বের করার একটি পদ্ধতি মাত্র।



গ্রাফ 13.8.1

ওপরের গ্রাফটি (গ্রাফ 13.8.1) হচ্ছে এলবো মেথডের গ্রাফ। এটি যেভাবে তৈরি করা হয়েছে তা হলো, প্রথমে K-এর বিভিন্ন মান নেওয়া হয়েছে। ধরি, যদি $K = 3$ হয়, তবে গোটা ডেটাসেট তিনটি ক্লাস্টারে ভাগ হয়ে যাবে, তাই না? এর পরে, আমাদের ডেটাসেটের প্রতিটি পয়েন্টকে এই তিনটি ক্লাস্টারের কোনো একটিতে অ্যাসাইন করা হয়েছে এবং প্রতিবার একটি করে ডেটা পয়েন্টকে কোনো ক্লাস্টারে অ্যাসাইন করার পরে সেই ক্লাস্টারের সেন্ট্রয়েড আপডেট করা হয়েছে (বিস্তারিত অধ্যায় ৭-এ রয়েছে)।

এরপরে, প্রতিটি ডেটাপয়েন্ট p যে ক্লাস্টারে অ্যাসাইন করা হয়েছে, সেই ক্লাস্টারের সেন্ট্রয়েডের সঙ্গে ওই ডেটা পয়েন্টের দূরত্ব নির্ণয় করা হয়েছে। অর্থাৎ, সহজ কথায় কোনো $p(x, y)$ যদি ক্লাস্টার 1-এ থাকে, তাহলে ক্লাস্টার 1-এর সেন্ট্রয়েড c_1 -এর সঙ্গে p -এর দূরত্ব বের করা হয়েছে। আবার, যদি অন্য একটি ডেটা পয়েন্ট $q(x, y)$ ক্লাস্টার 2-তে থাকে, তাহলে q -এর সঙ্গে ক্লাস্টার 2-এর সেন্ট্রয়েড c_2 -এর দূরত্ব বের করা হয়েছে। পরবর্তী সময়ে প্রতিটি ডেটা পয়েন্টের জন্য পাওয়া দূরত্বের মানকে বর্গ করে, বর্গগুলো যোগ করে আমরা যে মান পাই সেটি হচ্ছে $K = 3$ -এর জন্য সাম অব স্কয়ারড এরর (Sum of Squared Error) বা এসএসই (SSE) মান।

একইভাবে আমরা, K-এর বাকি সব মানের জন্য SSE মান বের করব। আমরা K-এর মান 1 থেকে 10 পর্যন্ত নিতে পারি। গ্রাফ 13.8.1-এ $K = 1$ থেকে 7 পর্যন্ত দেখানো হয়েছে। একটু লক্ষ করলে দেখব, গ্রাফটি 'দেখতে অনেকটা মানুষের হাতের কনুই' বা এলবো (Elbow)-এর মতো। গ্রাফে

অধ্যায় ১৩ : পারফরম্যান্স (Performance)

এলবো পয়েন্ট (elbow point) হিসেবে যেটিকে চিহ্নিত করা আছে, সেই পয়েন্টটিই হচ্ছে আমাদের হাতের কনুইয়ের ভাঁজ।

এই এলবো পয়েন্টটি বের করতে পারলেই আমাদের কাজ শেষ। K -এর যে মানের জন্য আমরা এই এলবো পয়েন্টটি পাচ্ছি, সেটি-ই হবে আমাদের K -এর অপটিমাল মান।

এখন, এই এলবো পয়েন্টের মান বের করার জন্য কিন্তু কোনো গাণিতিক সূত্র নেই। এটি চোখে দেখে অনেকটা আন্দাজ করে বের করতে হয়। লক্ষ করলে দেখবেন, গ্রাফটি $K = 1$ থেকে যখন শুরু হয় তখন SSE অনেক বেশি ছিল। K -এর মান বাড়ার সঙ্গে সঙ্গে হঠাতেও করে SSE-এর মান অনেকখানি নেমে আসে, একটি নির্দিষ্ট বিন্দু বা পয়েন্টের পরে এই পরিবর্তনের হার অনেকখানি কমে আসে। এই পয়েন্টটিকেই আমরা এলবো পয়েন্ট বলি।

অধ্যায় ১৪ : পরবর্তী সময়ে কী থাকছে

এই বইটিতে মূলত একেবারে হাতে-কলমে উদাহরণ দিয়ে বোঝানোর চেষ্টা করা হলো কীভাবে একটি মেশিন লার্নিং অ্যালগরিদম প্রয়োগ করতে হয়। কেউ যদি এখন একটু কষ্ট করে হলেও নিজের হাতে এই অ্যালগরিদমগুলোর কোড করে ফেলেন (পাইথনে হলে সুবিধা বেশি) তাহলে পুরো ব্যাপারটি একদম পানির মতো পরিষ্কার হয়ে যাবে। আমি এই বইতে কোনো কোড যুক্ত করিনি, কোডিংয়ের স্বাধীনতা পাঠকদের কাছেই রেখে দিলাম। এর কারণ হচ্ছে, আমার বইতে লিখে দেওয়া কোড দেখে দেখে টাইপ করে আসলে খুব বেশি কিছু বোঝা যাবে না। তার চেয়ে ভালো হয় যদি নিজে নিজে চিন্তা করে ধাপে ধাপে এই অ্যালগরিদমগুলো কোড করা যায়। কীভাবে কী করতে হবে সব তো দিয়েই দিয়েছি।

মেশিন লার্নিংয়ের এই অ্যালগরিদমগুলো আমি যেভাবে বইতে দেখিয়েছি, এইভাবে ম্যানুয়ালি না করে সরাসরি লাইব্রেরি ফাংশন ব্যবহার করেও করা যায় দু-তিন লাইনে। কিন্তু তাতে আসলে লাভের লাভ কিছু হবে না, ক্ষতিই হবে। আমরা যখন কোনো কিছু শিখি, তখন একেবারে গোড়া থেকে হাতে-কলমে শেখাই ভালো। এরপর একবার হাত পাকা হয়ে গেলে তখন আর লাইব্রেরি ফাংশন ব্যবহার করতে কোনো সমস্যা নেই, ওগুলো তো তৈরি করা হয়েছে ব্যবহার করার জন্যই। অনেকেই বলবেন যে চাকা বারবার আবিষ্কার করার কী দরকার? লাইব্রেরিতে তো কোড করাই আছে, সেটি ব্যবহার করলেই হয়! আবার কেন নিজে করব? এর উত্তর হচ্ছে, ওই যে বললাম, নিজের শেখার ভিত্তি পাকাপোক্ত করার জন্য। মেশিন লার্নিং শেখার জন্য হাতে-কলমে নিজে কোড করে করে বোঝার কোনো বিকল্প নেই।

আমি এখানে মেশিন লার্নিং অ্যালগরিদমগুলো কীভাবে পাইথনের Scikit Learn লাইব্রেরি ব্যবহার করে কোড করতে হয় সেই কোডগুলোর ডকুমেন্টেশনের লিংক দিয়ে দিচ্ছি :

- লিনিয়ার রিগ্রেশন

http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html

- লজিস্টিক রিগ্রেশন

http://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html

- সাপোর্ট ভেস্টের মেশিন

অধ্যায় ১৪ : পরবর্তী সময়ে কী থাকছে

<http://scikit-learn.org/stable/modules/svm.html>

- কে-নিয়ারেস্ট নেইবর

<http://scikit-learn.org/stable/modules/neighbors.html>

- কে-মিনস ক্লাস্টারিং

<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

- ডিসিশন ট্রি

<http://scikit-learn.org/stable/modules/tree.html>

- প্রিমিপাল কম্পোনেন্ট অ্যানালাইসিস

<http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>

- পারসেপ্ট্রন

http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html

- নিউরাল নেটওয়ার্ক

http://scikit-learn.org/stable/modules/neural_networks_supervised.html

এই কোডগুলো সব লাইব্রেরি ফাংশন ব্যবহার করে করা। আমার পরামর্শ থাকবে, আগে নিজের হাতে ম্যানুয়ালি অ্যালগরিদমগুলো কোড করে জিনিসগুলো বোবা, আমি অনেক খুঁটিনাটি বিষয় হয়তো বইটি যতটা সন্তুষ্ট সহজ রাখার স্বার্থে এড়িয়ে গিয়েছি। যেগুলো একেবারে না হলেই নয় সেগুলো দেখিয়েছি। তাই কোড করার সময় ইন্টারনেট থেকে আরো খুঁটিনাটি বিষয়গুলো দেখে নিতে হবে। নিজে নিজে পড়ার কোনো বিকল্প নেই।

এ বইটি শেষ করার পরে আমার পরামর্শ থাকবে পাইথন দিয়ে এই অ্যালগরিদমগুলোর কোড খুব ভালোমতো মকশো করে ফেলা। ম্যানুয়ালি ও লাইব্রেরি ফাংশন ব্যবহার করে, দুটিতেই যেন হাত

মেশিন লার্নিং অ্যালগরিদম

চালু থাকে। সেই সঙ্গে পাইথন ব্যবহার করে ডেটা ম্যানিপুলেশন, ডেটা রিপ্রেজেন্টেশন (NumPy, Pandas, Matplotlib মডিউল তিনটি খুব ভালোভাবে শিখতে হবে) শিখে নিতে হবে।

এগুলো খুব ভালোভাবে শেষ করতে পারলে, এরপর হাত পাকাতে হবে নিউরাল নেটওয়ার্ক ও ডিপ লার্নিংয়ে। আমার পরবর্তী বইটি হবে নিউরাল নেটওয়ার্ক ও ডিপ লার্নিংয়ের নানান বিষয় নিয়ে। সেটিও যখন আমরা মোটামুটি ভালোভাবে শেষ করতে পারব, তারপরে আসবে সবশেষের অংশ – হাতে-কলমে প্রোজেক্ট করা। আমি শুরুতেই হাতে-কলমে প্রোজেক্ট দিয়ে বোঝাতে পারতাম, কিন্তু তাতে বেসিকের অনেক ঘাটতি থেকে যেত।

আমি মনে করি, একটি সমস্যা সমাধানের ক্ষেত্রে কী হচ্ছে, কীভাবে হচ্ছে, কেন হচ্ছে সেটি কিছুটা অন্তত না বুঝে আসলে অন্তের মতো প্রোজেক্টে ঝাঁপিয়ে পড়াটা মোটেও কোনো কাজের কাজ নয়। প্রোজেক্ট করার আগে কিছু বেসিক বিষয় অবশ্যই জানা থাকতে হবে, তাহলেই কেবল প্রোজেক্টে কোনটি ব্যবহার করব এবং কেন করব, সেগুলো ভালোমতো বোঝা যাবে। আর তাই, আমার এই সিরিজের শেষ বইটি হবে মেশিন লার্নিংয়ের ৫টি প্রোজেক্ট নিয়ে। বইটি হবে পাইথনে, হাতে-কলমে একেবারে গোড়া থেকে কীভাবে একটি প্রোজেক্ট করতে হয় সেটি ওই বইয়ে শেখানো হবে।

আমাদের মেশিন লার্নিংয়ের অ্যালগরিদমগুলোতে হাতেখড়ির এইখানেই সমাপ্তি টানছি।

এক নজরে কিছু সূত্র

১। নিউমেরিকাল অ্যানালাইসিস ব্যবহার করে লিনিয়ার রিপ্রেশন

সরলরেখার সমীকরণ, $y = mX + c$

$$\text{যেখানে, } m = \frac{\bar{x} \cdot \bar{y} - \bar{x}\bar{y}}{(\bar{x})^2 - \bar{x}^2}$$

$$c = \bar{y} - m\bar{x}$$

২। গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ব্যবহার করে লিনিয়ার রিপ্রেশন

হাইপোথিসিস, $h_w(x) = w_0 + w_1x$

$$\text{কস্ট ফাংশন } J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

$$\text{অবজেকটিভ ফাংশন} = \underbrace{\min_{w_0, w_1}} \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

$$\text{প্যারামিটার আপডেট, } w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1)$$

$$w_0\text{-এর ক্ষেত্রে, } \frac{\partial}{\partial w_0} J(w_0, w_1) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}$$

$$w_1\text{-এর ক্ষেত্রে, } \frac{\partial}{\partial w_1} J(w_0, w_1) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x^{(i)}$$

৩। গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ব্যবহার করে মাল্টিভ্যারিয়েট লিনিয়ার রিপ্রেশন

হাইপোথিসিস, $h_w(x) = w_0x_0 + w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n$
 $= w^T x$

$$\text{যেখানে, } w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix} \text{ ও } x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

$$\text{কস্ট ফাংশন, } J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

$$\text{অবজেকটিভ ফাংশন} = \underbrace{\min_{w_0, w_1, w_2, \dots, w_n}} \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

$$\text{প্যারামিটার আপডেট, } w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1, w_2, \dots, w_n)$$

$$\frac{\partial}{\partial w_0} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}; \text{ যখন, } j = 0$$

$$\frac{\partial}{\partial w_j} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_j^{(i)}; \text{ যখন, } j = 1, \dots, n$$

৪। L2 রেগুলারাইজেশন

$$\text{কস্ট ফাংশন, } \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

৫। লজিস্টিক রিগ্রেশন

$$\text{সিগময়েড ফাংশন, } g(w^T x) = \frac{1}{1 + e^{-w^T x}}$$

$$\text{কস্ট ফাংশন, } J(W) = -\frac{1}{m} \left[\sum_{i=1}^m \left\{ y^{(i)} \log(h_w(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_w(x^{(i)})) \right\} \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

$$\text{প্যারামিটার আপডেট, } w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(W)$$

$$\frac{\partial}{\partial w_0} J(W) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_0^{(i)}; j = 0$$

$$\frac{\partial}{\partial w_j} J(W) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_j^{(i)} - \frac{\lambda}{m} w_j; j = 1, \dots, n$$

৬। ডিসিশন ট্রি

$$\text{এন্ট্রোপি, } Entropy(S) = \sum_{i=1}^n -P_i \log_2 P_i$$

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times Entropy(S_v)$$

৭। পারফরম্যান্স

$$\text{আর স্কয়ারড, } R^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

$$\text{অ্যাকুরেসি} = \frac{TP + TN}{TP + FP + FN + TN}$$

$$\text{প্রিসিশন} = \frac{TP}{TP + FP}$$

$$\text{রিকল} = \frac{TP}{TP + FN}$$

$$\text{এফ-ওয়ান} = \frac{2 * Precision * Recall}{Precision + Recall}$$

$$\text{মেজার} = \frac{TN}{Precision + Recall}$$

$$\text{স্পেসিফিসিটি} = \frac{TN}{TN + FP}$$