

সূচিপত্র

ভূমিকা	1.1
মেশিন লার্নিং পরিচিতি	1.2
মেশিন লার্নিং প্রস্তুতি	1.3
পাইথন প্যাকেজ ইন্সটলেশন	1.3.1
মেশিন লার্নিং পাইথন টুলস	1.4
IPython/ Jupyter Notebook পরিচিতি	1.4.1
মেশিন লার্নিং কাজের ধারা	1.5
কীভাবে সঠিক প্রশ্ন করতে হয়?	1.5.1
ডেটা প্রিপ্রেসিং - ১	1.5.2
ডেটা প্রিপ্রেসিং - শেষ পর্ব	1.5.3
অ্যালগরিদম সিলেকশন	1.5.4
মডেল ট্রেইনিং	1.5.5
মডেল পারফর্মেন্স টেস্টিং - ১	1.5.6
মডেল পারফর্মেন্স টেস্টিং - শেষ পর্ব	1.5.7
লিনিয়ার রিগ্রেশন	1.6
লিনিয়ার রিগ্রেশন প্রাথমিক আলোচনা	1.6.1
লিনিয়ার রিগ্রেশন পর্ব-২ ও গ্রেডিয়েন্ট ডিসেন্ট	1.6.2
মাল্টিভ্যারিয়েবল লিনিয়ার রিগ্রেশন	1.6.3
প্র্যাটিক্যাল লিনিয়ার রিগ্রেশন	1.6.4
মাল্টিভ্যারিয়েবল লিনিয়ার রিগ্রেশন : গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদমের রকমফের	1.6.5
প্র্যাটিক্যাল মাল্টিভ্যারিয়েবল লিনিয়ার রিগ্রেশন : গ্রেডিয়েন্ট ডিসেন্টের নরমাল ফর্ম	1.6.6
লজিস্টিক রিগ্রেশন	1.7
লজিস্টিক রিগ্রেশন : পরিচিতি	1.7.1
লজিস্টিক রিগ্রেশন : প্রব্যাবিলিটি, বানুলি ও বাইনোমিয়াল ডিস্ট্রিবিউশন	1.7.2
পরিশিষ্ট	1.8
Numpy পরিচিতি	1.8.1
মিনি প্রজেক্ট	1.9
হাতে লেখা বাংলা সংখ্যা রিকগনিশন	1.9.1
কৃত্রিম নিউরনে অনুরণন পর্ব-১	1.9.1.1
কৃত্রিম নিউরনে অনুরণন পর্ব-২	1.9.1.2

ভূমিকা

পাইথন, ম্যাটল্যাব ও জাভাস্ক্রিপ্টে প্রেডিষ্টিভ মডেল বিল্ডিং ও পারফর্মেন্স টেস্টিং

সংক্ষেপ

মেশিন লার্নিং কী তা নিয়ে পরে বিস্তারিত আলোচনা করা হবে তবে সংক্ষেপে বলা যেতে পারে, যদি কোন মেশিন অভিজ্ঞতার উপর ডিপ্টি করে নিজে নিজে শিখতে পারে কিংবা ডিবিস্যুনেটী করতে পারে তাহলে বলা যায় সিস্টেমটি ইন্টেলিজেন্ট বা ML Activated।

বর্তমানে যেকোন ইঞ্জিনিয়ারিং বিভাগের জন্য মেশিন লার্নিং একটি গুরুত্বপূর্ণ বিষয় হয়ে দাঁড়িয়েছে। ডেটা অ্যানালাইসিস, ক্লাসিফিকেশন, প্রেডিকশনের জন্য এটা শেখা অত্যন্ত জরুরি। বিগ ডেটা, ডেটা সায়েন্স, আর্টিফিশিয়াল ইন্টেলিজেন্সের সাথে মেশিন লার্নিং ও তপ্রোতভাবে জড়িত। বর্তমানে সাধারণ ওয়েব অ্যাপ কিংবা মোবাইল ফোনেও ML এর বিভিন্ন ফিল্ডে অ্যাপ্লাই করা হয় যাতে আপনার ব্যবহারকৃত অ্যাপ্লিকেশনটি আরও ইন্টেলিজেন্ট হয় এবং আপনার মনের কথা বোঝার ক্ষমতা অর্জন করতে পারে। সাধারণ অ্যাপ ও ML ইন্পুটেড অ্যাপের মধ্যে তফাও হল এই, সাধারণ অ্যাপ্লিকেশন সব সময় সাধারণই থাকবে কিন্তু ML ইন্পুটেড অ্যাপটি হবে অনন্যসাধারণ, প্রতিবার ব্যবহার করার পর আপনার মনে হবে অ্যাপটি যেন আরও ইন্টেলিজেন্ট হচ্ছে। তবে ML যে শুধু অ্যাপকে ইন্টেলিজেন্স দিতে পারে তাই নয়, রোগ নির্ণয় থেকে শুরু করে যেকোন ধরণের ক্লাসিফিকেশন ও প্রেডিকশনের জন্য ML এর জুড়ি নেই। এই বইয়ে মূলত মডেল তৈরির পশাপাশি এর পিছনের ম্যাথমেটিক্সেরও ব্যাখ্যা যথাসাধ্য সাবলীল ভাষায় উপস্থাপন করা হবে।

বইটি কাদের জন্য?

আর্টিফিশিয়াল ইন্টেলিজেন্স, বিগ ডেটা, ডেটা মাইনিং এ আগ্রহী কিংবা ML প্র্যাকটিশনার, ML হবিট ও ML বিগিনারদের জন্য এই বই। আর্টিফিশিয়াল ইন্টেলিজেন্স এর নাম শুনেছেন কিন্তু অ্যাপ্লাই করার যাদের শখ তারাও চাইলে বইটি পড়তে পারেন। বিস্তারিত নিচে বলা হল।

বইটি পড়া শুরু করার আগে যা যা জানা লাগবে (* চিহ্নিত টপিক আলোচনা বহির্ভূত থাকবে)

- বেসিক পাইথন প্রোগ্রামিং*
- বেসিক MATLAB প্রোগ্রামিং*
- বেসিক JavaScript প্রোগ্রামিং*
- লিনিয়ার অ্যালজেব্রা*
- Pythonic Syntactic Sugar

- OOP Python পারলে স্টোকে প্লাস পয়েট হিসেবে ধরা যাবে
- ক্যালকুলাস (ইন্টিগ্রেল ও ডিফারেনশিয়াল)
- বেসিক পরিসংখ্যান জ্ঞান যেমন: Mean, Mode, Median, Variance, Co-Variance, Correlation, Standard Deviation...

কী কী আলোচনা করা হবে এই বইয়ে?

মেশিন লার্নিং আসলে অনেক বিস্তৃত একটি বিষয়। একটি বইয়ে এটা কম্পিউট করা সম্ভব নয়। প্রতিনিয়তই ভাল থেকে আরও ভাল মডেল বিন্দু করার পদ্ধতির রিসার্চ চলছে। এই বইয়ে মূলত আপনাকে মেশিন লার্নিংয়ের সাথে পরিচয় করিয়ে দেওয়া হবে, তবে অ্যাডভান্সড লেভেলে যেতে হবে আপনার নিজেরই। তাহলে টপিকগুলো এক নজরে দেখা যাক (সম্পূর্ণ টপিক পরে আপডেট করা হবে):

- প্রয়োজনীয় সফটওয়্যার ইন্টলেশন
 - Anaconda Python Distribution ইন্টলেশন
 - PyCharm IDE এর সাথে পরিচয় ও ইন্টলেশন
 - Sublime Text 3 কে Python এর উপযোগী করে তোলা
- মেশিন লার্নিং কিক স্টার্ট
 - মেশিন লার্নিং কী?
 - মেশিন লার্নিংয়ের প্রয়োগ কী?
 - রিগ্রেশন কী?
 - লিনিয়ার ও পলিনমিয়াল রিগ্রেশন কী?
 - সিম্পল লিনিয়ার রিগ্রেশন এর মাধ্যমে প্রেডিকশন (Sklearn মডিউল ব্যবহার করে)
 - সিম্পল লিনিয়ার রিগ্রেশন এর মাধ্যমে প্রেডিকশন (Scratch থেকে মডেল তৈরি করা)
- মেশিন লার্নিং কিক স্টার্ট ২
 - Supervised Learning
 - Unsupervised Learning
- দুইটা প্রয়োজনীয় প্রেডিকশন অ্যালগরিদম
 - কেন এই দুইটা অ্যালগরিদম প্রয়োজনীয়?
 - পেনালাইজড রিগ্রেশন মেথড (Penalized Regression Method) কী?
 - এনসেম্বল মেথড (Ensemble Method) কী?
 - কীভাবে অ্যালগরিদম সিলেক্ট করবেন?
 - প্রেডিষ্টিভ মডেল তৈরি করার সাধারণ বেসিপি
- সমস্যা চিনুন ডেটাসেট চেনার মাধ্যমে
 - নতুন কোন সমস্যাৰ ব্যবচ্ছেদ
 - অ্যাট্ৰিবিউট ও লেবেল কী? সমার্থক শব্দগুলো কী কী?
 - ডেটাসেট এর যেসব জিনিসেৰ দিকে খেয়াল ৱাখতে হবে
- মডেল ও Cost Function

- মডেল রিপ্রেজেন্টেশন
 - Cost Function
 - Cost Function Intuition - 1
 - Cost Function Intuition - 2
 - Ovefitting - আপনার বানানো মডেল কী একটু বেশি ভাল পার্ফর্ম করছে?
- Parameter লার্নিং
 - গ্রেডিয়েন্ট ডিসেন্ট
 - গ্রেডিয়েন্ট ডিসেন্ট ইন্টুইশন
 - লিনিয়ার রিগ্রেশনে গ্রেডিয়েন্ট ডিসেন্ট
 - চলবে

সচরাচর জিজ্ঞাস্য প্রশ্ন:

মেশিন লার্নিং আমার Career এ কী কাজে লাগবে?

মেশিন লার্নিং খুবই বিস্তৃত একটি এরিয়া, আর্টিফিশিয়াল ইন্টেলিজেন্স থেকে প্যাটার্ন রিকগনিশন এর অন্তর্গত। প্রতিদিনই প্রচুর পরিমাণ ডেটা নিয়ে কাজ চলে। প্যাটার্ন রিকগনিশনের মাধ্যমে এই ডেটাকে গুগল, মাইক্রোসফটের মত বড় বড় কোম্পানি প্রসেস করে। এই কারণেই গুগল সার্চ দিতে এত আরাম। যত ভুলই থাকুক না কেন, সে সেটাকে ঠিক করে নেয়, ধৰা যাক আপনি নিয়মিত প্রোগ্রামিং এর উপরে ডিডিও দেখেন ইউটিউবে। বেশ কিছুদিন দেখলে সে এমন এমন সব ডিডিও রিকমেন্ডেশনে দেবে যে মনে হবে এই ডিডিওটাই যেন আপনি চাইছিলেন।

কেরিয়ারে লাগবে কী লাগবে না সেটা আপনার ব্যাপার। আপনি যদি ডাক্তার হন, হাস্পাতাল পাতলা প্রোগ্রামিং পারেন, কিছুটা ML, কিছুটা Data Science এবং কিছুটা NLP (Natural Language Processing) বা NLU (Natural Language Understanding) এর মাধ্যমে বানাতে পারেন আর্টিফিশিয়াল রেইন যেটা হ্যত রোগের লক্ষণ ও রোগ ইনপুট নিতে পারে এবং আউটপুটে প্রতিষেধক দিতে পারে। আপনি যখন কোথাও ঘুরতে যাবেন, চ্যাটবট হিসেবে আপনার তৈরি করা রেইন ই ডাক্তার হিসেবে ছোটখাট রোগের চিকিৎসা করতে পারবে।

কেরিয়ারে লাগুক বা না লাগুক, CS এর একটি বিশাল ইন্টারেস্টিং এরিয়া হল ML। কমবেশি সবারই ML এ ব্যবহৃত কিওয়ার্ডগুলো জানা উচিত।

মেশিন লার্নিং কাদের জন্য?

মেশিন লার্নিং শেখার জন্য সায়েন্স ব্যাকগ্রাউন্ড হলে খুবই ভাল। কেননা সাধারণ প্রোগ্রামিং করা হয় Explicit প্রোগ্রামিং এর মাধ্যমে কিন্তু প্রেডিকশনের ব্যাপার যেখানে জড়িত সেখানে Explicit প্রোগ্রামিংয়ের মাধ্যমে সে সমস্যা সল্ভ করা যায় না। যদি সায়েন্স সম্পর্কে বিদ্যুমাত্র আইডিয়া না থাকে তাহলে আন্ডারলাইং কনসেপ্টগুলো বুঝতে সমস্যা হতে পারে তবে, ম্যাথ বাদে মডেল ডেভেলপ করতে পারবেন, কিন্তু মডেলের যে অপ্টিমাইজেশন, সেটা ম্যাথ ছাড়া করা অসম্ভবের কাছাকাছি।

কখন মেশিন লার্নিং ব্যবহার করা উচিত?

যদি মনে হয় আপনার অ্যাপে মিডিজিক/ডিডিও/ঁগ পোস্ট বিকমেন্ডেশন সেট করা প্রয়োজন। কিংবা আপনার ওয়েবসাইটে স্মার্ট স্প্যামার ব্রকার প্রয়োজন। কিংবা কোন কোন প্যারামিটারের উপর ভিত্তি করে আপনার ওয়েবসাইটে কেউ Ad এ ক্লিক করে ... ইত্যাদি।

এই বইটি পড়ার পূর্বশর্ত কী?

[উপরে বলা আছে](#)

এতগুলো ল্যাঙ্গুয়েজ নিয়ে আলোচনার কারণ কী?

যদি একজন ফুল স্ট্যাক জাভাস্ক্রিপ্ট ডেভেলপার তার ওয়েব অ্যাপে ML মেথড অ্যাপ্লাই করতে চাইলে তাকে নতুন করে Python শিখতে হবে, এইসব বামেলা এডানোর জন্য একই জিনিস ভিন্ন ভিন্ন প্ল্যাটফর্মে অ্যাপ্লাই করে দেখানো হবে।

কোন কোন বই ফলো করা হবে?

- Machine Learning in Python : Essential Techniques for Predictive Analysis [Wiley] - Michael Bowles
- Mastering Machine Learning with Scikit-Learn [PACKT]
- Data Science from Scratch [OREILY] - Joel Grus
- Building Machine Learning System with Python [PACKT]

মেশিন লার্নিং নিয়ে কোন কোন টিপ্পি সিরিজ বানানো হয়েছে?

মেশিন লার্নিং ব্যাপারটা জটিল ও কাঠখোটা লাগলেও, কোন কিছু শেখার সময় শুধু থিওরি জানলে সেটা জটিল ও নিরস থেকে যায়। কিন্তু আমরা যদি পশাপাশি ও টপিক রিলেটেড মুভি বা সিরিজ দেখি তাহলে আমাদের আগ্রহ বহুগুণ বেড়ে যায়। সেজন্য এই সংক্ষিপ্ত লিস্ট।

Person Of Interest

মেশিন লার্নিং কে বেজড করে চমৎকার উপভোগ্য একটি টিপ্পি সিরিজ, মেশিন লার্নিং কে ভালবাসার জন্য এই একটাই যথেষ্ট। এর মূল চরিত্রে থাকে চরম প্রতিভাবান প্রোগ্রামার Harold Finch ও তার ডান হাত John Reese। Harold Finch এমন একটি মেশিন তৈরি করেন যেটা কোন দৃঘটনা ঘটার আগেই প্রেডিক্ষন করতে পারে এবং Harold Finch এর কাজ হল সেই দৃঘটনা প্রতিরোধ করা।

এটাতে দেখানো হয়েছে

- Natural Language Understanding (যেখানে Harold তার এই Machine এর সাথে English ল্যাঙ্গুয়েজের মাধ্যমে কমিউনিকেট করে)
- Image Processing (Facial Recognition, Object Recognition, Optical Character Recognition ...)
- Artificial Neural Network: প্রায়ই দেখা যায় বেশকিছু ছবি লাইনের মাধ্যমে ইন্টারকানেক্টেড, এগুলো দিয়ে আসলে Artificial Neuron এর কানেকশন বোঝানো হয়েছে। এই বইয়ের একটি বিশাল অংশ জুড়ে থাকবে ANN।

Silicon Valley

সিরিজটি মূলত প্রতিভাবান প্রোগ্রামার ও তার ডেটা কম্প্রেশন কোম্পানির কাহিনী নিয়ে তবুও এখানে ML এর প্রয়োগটা ৩য় সিজনে বলা হয়।

ডেটা কম্প্রেশন অ্যালগরিদমের মূল কাজ থাকে কোন একটা ডেটাসেটে Information কতটা থাকে? যদি অ্যালগরিদম ডিটেক্ট করতে পারে যে Dataset এর একটা নির্দিষ্ট অংশ Redundant মানে, সেটা মুছে দিলেও ক্ষতি নেই। সেই অংশটুকু বাদ দিলে কম্প্রেসড ডেটার সাইজ আগের চেয়ে কম হবে সেটাই স্বাভাবিক। কিন্তু Information extraction টাই হল আসল চ্যালেঞ্জ।

ধরুন, আপনার ক্লাসের শিক্ষক ক্লাসে শুধু 'ক' শব্দটি উচ্চারণ করেন, এটা থেকে বুঝা যায় যদিও বা 'ক' এর সমষ্টিগুলো ডেটাসেট হিসেবে গ্রহণযোগ্য কিন্তু এতে Information এর পরিমাণ 0। আমরা এই সমস্ত 'ক' এর স্ট্রিং নিয়ে কম্প্রেস করলে আউটপুট ফাইলের সাইজ হবে ০ বাইট। যেহেতু এতে আরো কোন Information নাই। কিন্তু বাজে অ্যালগরিদম অ্যাপ্লাই করলে আউটপুট ফাইলের সাইজ ইনপুটের সমান বা কিছুটা কম হতে পারে।

মেশিন লার্নিংয়ের অন্যতম অ্যাপ্লিকেশন প্রেডিক্ষন করা। তাই ডেটা কম্প্রেশনে এটা ব্যবহার করে আমরা অতি সহজেই Information extract করতে পারি। কিন্তু আমাদের মডেলের পার্ফর্মেন্স যদি খারাপ হয় সেক্ষেত্রে AI সিস্টেমটা Redundant অংশ রেখে Information কেটে দিতে পারে।

৩য় সিজনে (নন স্পয়লার) দেখা যায় কোন একটা পরিস্থিতিতে Richard কে বলা হয় মেশিন লার্নিং সিস্টেম ফেলে দিতে, কিন্তু সে বলে তাতে তার কম্প্রেশন অ্যালগরিদম ইউজলেস হয়ে যাবে।

আমরা ধারণা করতে পারি এখান থেকে ML মেথডলজি অ্যাপ্লাই করে Information Extraction ই ছিল Middle Out (কাল্পনিক অ্যালগরিদম) এর মূল কাজ।

অত্যন্ত মজার ও Insightful একটি টিপ্পি সিরিজ Silicon Valley। হ্যাত ML এর সাথে পুরোপুরি যুক্ত না থাকলেও এর কাহিনীগুলো আপনার সময় ভালভাবে কাটাতে সাহায্য করবে।

লেখক সম্পর্ক

মানস

Embedded System Engineer

AplombTech BD Ltd.

[ওয়েবসাইট](#) | [ব্লগ](#) | [ফেসবুক](#) | [ইমেইল](#) | [গিটহাব](#)

সম্পূর্ণ বিনামূল্যে বইটি যে কেউ পড়তে পারেন ও কন্টেন্ট শেয়ার করতে পারেন কিন্তু মূল কন্টেন্ট অবিকৃত রেখে
এবং পর্যাপ্ত ক্রেডিট দিয়ে। কমার্শিয়াল ব্যবহার সম্পূর্ণ নিষিদ্ধ।



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](#).

মেশিন লার্নিং পরিচিতি

মেশিন লার্নিং কী?

মেশিন লার্নিং শুরু করার আগে কয়েকটি কেতাবি সংজ্ঞা দেখা যাক। এই ব্যাপারে **Arthur Samuel** বলেন,

Field of study that gives computers the ability to learn without being explicitly programmed.

অর্থাৎ কিনা, কম্পিউটারের যদি এমন কোন অলৌকিক ক্ষমতা থাকে যার জন্য সে যেকোন কিছু আগে থেকে ওঁ বিষয়ক প্রোগ্রাম লেখা ছাড়াই শিখতে পারে।

ধরা যাক, একটা বাইপেডাল (হিউম্যানয়েড বা দুই পা ওয়ালা) ৰোবট যদি নিজে হাঁটা শিখতে পারে কোন নির্দিষ্ট হাঁটার প্রোগ্রাম ছাড়াই তবে বলা যাবে ৰোবটে লার্নিং অ্যালগরিদম ব্যবহার করা হয়েছে। আমরা একটা বাইপেডাল ৰোবটের হাঁটার জন্য সহজেই প্রোগ্রাম লিখে দিতে পারি। কিন্তু সেই হাঁটাকে ইটেলিজেন্ট বলা যাবে না কোনভাবেই, একটা এমবেডেড সিস্টেম যে জন্য প্রোগ্রাম করা হয় সে যদি শুধু ওঁ নির্দিষ্ট কাজটাই করে তাহলে সেটা ইটেলিজেন্ট কীভাবে? পরিবর্তনের সাথে যদি ডিভাইসের আচরণ পরিবর্তিত হয় তাহলেই তাকে ইটেলিজেন্ট বলা যেতে পারে।

Tom Michel এর মতে,

A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**.

হঠাতে করে সংজ্ঞাটা দেখলে একটু সমস্যা হতে পারে, তাই একে একটা উদাহরণের মাধ্যমে বলা যেতে পারে,

ধরি, আমি এমন একটি মেশিন তৈরি করলাম যে দাবা (Chess) খেলতে পারে, তাহলে নিচের প্যারামিটারগুলোকে আমরা এভাবে লিখতে পারি,

E = ধরি মেশিনটা ৫০০ টা কম্পিউট সেট দাবা খেলল

T = দাবা খেলাটাই মেশিনের **Task**

P = মেশিন খেলায় জিতল না হারল

সংজ্ঞানুযায়ী,

যদি মেশিনের খেলার সংখ্যার বৃদ্ধির (**E**) পাশাপাশি তার জেতার হার বেড়ে যায় (**P**) তাহলে বুঝতে হবে সেই মেশিন আসলেই শিখছে।

আর এটা Explicitly প্রোগ্রামের মাধ্যমে করা নিত্যতই অসম্ভব।

ডেটা সায়েন্স, আর্টিফিশিয়াল ইটেলিজেন্স ও মেশিন লার্নিং

ডেটা সায়েন্স (Data Science)

ডেটা সায়েন্স আসলে পরিসংখ্যান, মেশিন লার্নিং ও ডেটা ভিজুয়ালাইজেশন এর সমষ্টি। একজন ডেটা সায়েন্টিস্টের কাজ হচ্ছে ডেটাসেট এর মাধ্যমে কিছু প্রশ্নের উত্তর খোঁজা। [বিস্তারিত](#)।

আর্টিফিশিয়াল ইন্টেলিজেন্স

আর্টিফিশিয়াল ইন্টেলিজেন্স কিছু সমস্যা ও সমস্যা সমাধানের পদ্ধতির সমষ্টি যেগুলো ব্যবহার করে জটিল সমস্যা সমাধান করা যায়। কম্পিউটারকে তাস, দাবা খেলা শেখানো, ন্যাচারাল ল্যাঙুয়েজ ট্রান্সলেশন, সিকিউরিটি স্ট্র্যাটেজি ম্যানেজমেন্ট AI অঙ্গর্গত। AI এর প্রবলেম যে বাস্তব ডেটাসেট বেজড হতে হবে এমন কোন কথা নাই, যিওরিটিক্যাল হতে পারে।

মেশিন লার্নিং

মেশিন লার্নিং আর্টিফিশিয়াল ইন্টেলিজেন্সের একটা বিভাগ যেখানে ইন্টেলিজেন্ট সিস্টেম তৈরি করা হয় ডেটাসেট কিংবা ইন্টারঅ্যাক্টিভ এক্সপেরিয়েন্সের মাধ্যমে। মেশিন লার্নিং টেকনোলজি Cybersecurity, Bioinformatics, Natural Language Processing, Computer Vision, Robotics ছাড়াও প্রচুর ক্ষেত্রে ব্যবহার করা হচ্ছে।

Machine Learning এর সবচেয়ে বেসিক কাজ হল ডেটা ল্যাসিফিকেশন, যেমন একটা ই-মেইল বা ওয়েবসাইটের কমেন্ট স্প্যাম কিনা তা চেক করা। বর্তমানে Deep Learning বা Deep Network এর উপরে প্রচুর রিসার্চ চলছে, মূলত Convolution Neural Network এসব ক্ষেত্রে ব্যবহার করা হয়।

বর্তমানে ইন্ডাস্ট্রিয়াল লেভেলে মেশিন লার্নিং অত্যন্ত গুরুত্বপূর্ণ একটা বিষয়। কিছু না কিছু মেশিন লার্নিং মেথডলজি জানা সকলেরই উচিত। মেশিন লার্নিংয়ের বেশ কিছু জিনিসই ডেটা সায়েন্সের সাথে ওভারল্যাপ করবে, কিন্তু মেশিন লার্নিংয়ের মূল টাগেট হল প্রেডিক্টিভ মডেল বিন্দু করা।

একনজরে

অর্থাৎ, AI ইন্টেলিজেন্ট মেশিন তৈরিতে সাহায্য করে, ML হল AI এর সাবফিল্ড যেটা মেশিনকে কোন কিছু শিখতে সাহায্য করে এবং সর্বশেষ ডেটা সায়েন্স লার্নিং অ্যালগরিদম বেজড মেশিনকে সাহায্য করে ডেটা প্যাটার্ন বের করতে যা সে পরবর্তী কাজে ব্যবহার করতে পারবে।

Data Science, ML ও AI কে প্রায়ই একই জিনিস মনে হবে কারণ এগুলার মধ্যে পার্থক্য খুবই অল্প। তবে ডেটা সায়েন্স সম্পর্কে একটা প্রচলিত কৌতুক হল,

একজন ডেটা সায়েন্টিস্টের কম্পিউটার জ্ঞান একজন পরিসংখ্যানবিদের চেয়ে বেশি এবং তার পরিসংখ্যান জ্ঞান একজন কম্পিউটার সায়েন্টিস্টের চেয়ে বেশি।

লার্নিং অ্যালগরিদম এর প্রকারভেদ

সুপারভাইজড লার্নিং (মেশিন কে শেখানো)

প্রথমে মেশিনকে শেখাবেন, তারপর তার শিক্ষাকে কাজে লাগাবেন

আনসুপারভাইজড লার্নিং (মেশিন নিজে নিজেই কিছু করা শিখবে)

মেশিন নিজে নিজেই কিছু করা শিখবে তারপর সেই শিক্ষা কাজে লাগিয়ে ডেটা স্ট্রাকচার ও প্যাটার্ন বুঝবে যেটা কিনা আপনিও বোঝেন না

সুপারভাইজড লার্নিং (পরিচিতি)

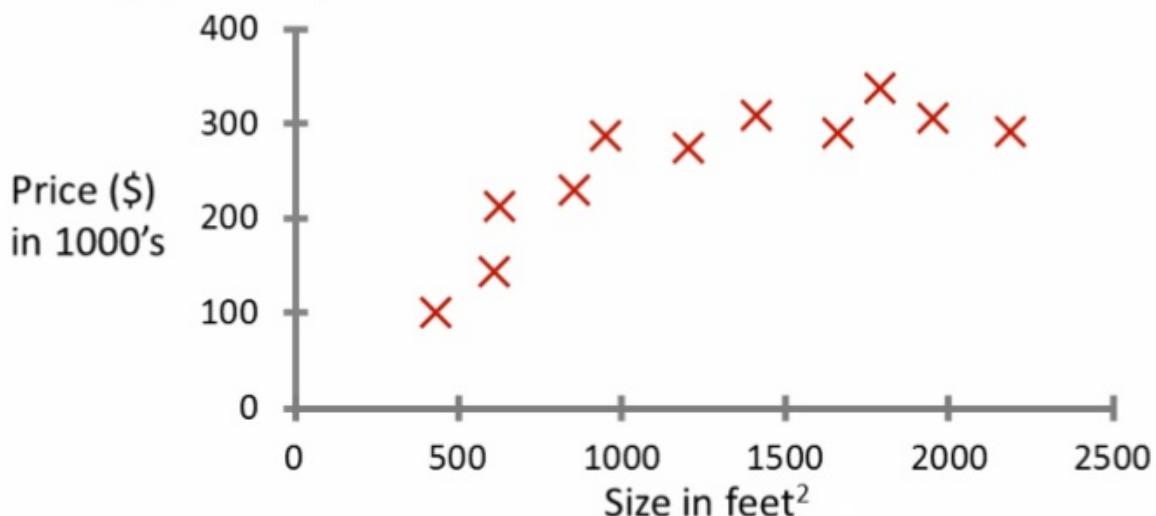
Regression

ML এর সবচেয়ে পরিচিত সমস্যা নিয়ে আলোচনা শুরু করা যাক। ধরুন আমার কাছে কিছু ডেটা আছে, ঘরের সাইজ ও তার দরদাম।

এই ডেটাসেট আমি যদি প্লট করি তাহলে নিচের মত একটি গ্রাফ দেখতে পাব।

ডেটাসেট:

Housing price prediction.



এই সমস্যার ছবি নেওয়া হয়েছে Andrew Ng এর মেশিন লার্নিং কোর্সের রিসোর্স থেকে। সরাসরি ব্যবহার করা হল।

সমস্যা:

উপর্যুক্ত ডেটাসেট দিয়ে আমাকে বের করতে বলা হল,

- যদি তোমার বন্ধুর বাড়ির সাইজ 750 square feet হয়, তাহলে দাম কত?

সমাধান:

আমি যদি এমন একটা ইকুয়েশন বের করতে পারি যেটাতে Area বসালে Corresponding প্রাইস পাওয়া যায়, তার মানে

$$y = f(x)$$

বা,

$$\text{Price} = f(\text{Area})$$

তার মানে আমাদের বের করতে হবে $f()$ ফাংশনটা আসলে কী? আমি এখানে বলব না কীভাবে $f()$ বের করতে হবে।

সমস্যা থেকে যেসব তথ্য আমরা পাই

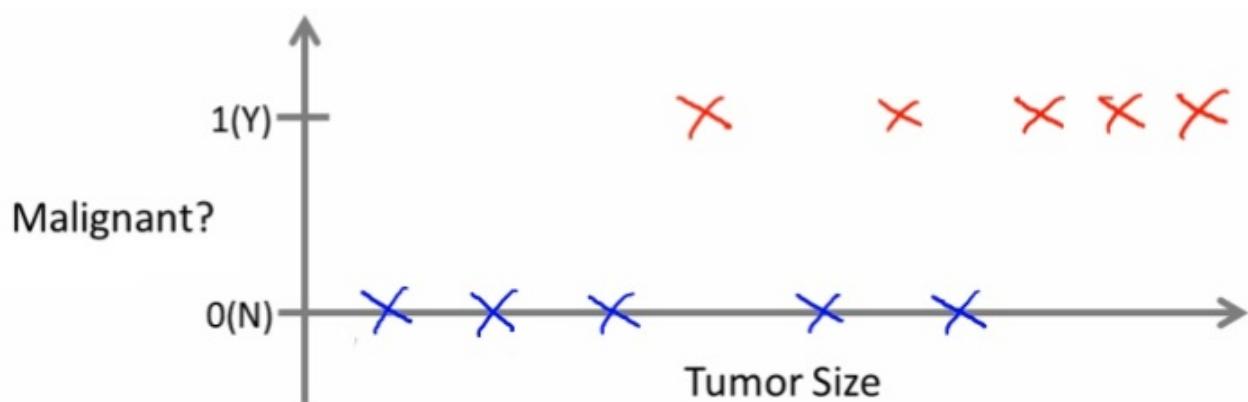
- এখানে আমরা আমাদের অ্যালগরিদমকে একটা ডেটাসেট দিচ্ছি যেখানে 'সঠিক উত্তর' দেওয়া হয়েছে। (গ্রাফ থেকে)
- তার মানে আমরা নির্দিষ্ট সাইজের বাড়ির আসল দাম জানি
 - এই ডেটা আমরা অ্যালগরিদমে ফিড করার মাধ্যমে ওকে শেখাতে পারি যে এই সাইজের বাড়ির দাম হ্য এত। একে Training Data বলা হ্য।
 - এবাব এই Training Data এর উপর ভিত্তি করে আমরা এমন সাইজের বাড়ির দাম জানতে পারি যে সাইজটি Training Data তে ছিল না। যেমন আমি যদি 3000 sq ft এর বাড়ির দাম জানতে চাই সেটা কিন্তু ডেটাসেট এ নেই! কিন্তু আমার তৈরি করা মডেল আগের Experience এর উপর ভিত্তি করে 3000 sq ft বাড়ির দাম অনুমান (Prediction) করতে পারে

এই সমস্যাটি **Regression** সমস্যার অন্তর্গত

কাবণ, পূর্বের ব্যবহৃত মান থেকে আমরা নতুন একটি মান অনুমান করার চেষ্টা করছি। পূরবতী উদাহরণ দেখলে পরিষ্কার হবে।

Classification

এবাব আরেক ধরণের ডেটাসেট দেখা যাক, যেখানে টিউমারের আকারের সাথে সাথে বলা আছে ও আকারের টিউমারটি Deadly, Malignant বা প্রাণঘাতী কিনা।



এখনে 1 কে হ্যাঁ হিসেবে ধরা হয়েছে এবং 0 কে না হিসেবে।

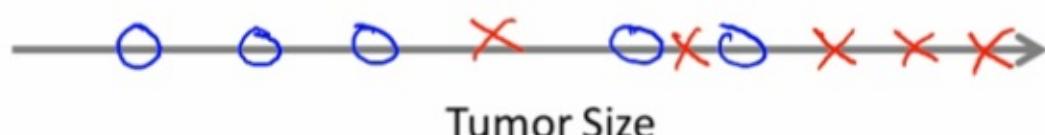
সাধারণ জ্ঞান অনুসারে বলা যায় যে টিউমারের যদি আকারে বড় হয় তবে তার প্রাণঘাতী হওয়ার সম্ভাবনা বেড়ে যায়।

কিন্তু ডেটাসেট থেকে দেখা যাচ্ছে যে কিছু কিছু টিউমার আকারে বড় হলেও প্রাণঘাতী নাও হতে পারে। আবার কিছু কিছু ছোট টিউমারও প্রাণঘাতী হতে পারে।

সমস্যা: আমাদের এমন একটি প্রে�িক্শন মডেল তৈরি করতে পারে যে বলতে পারবে টিউমার প্রাণঘাতী কিনা
(আকারের উপর ভিত্তি করে)

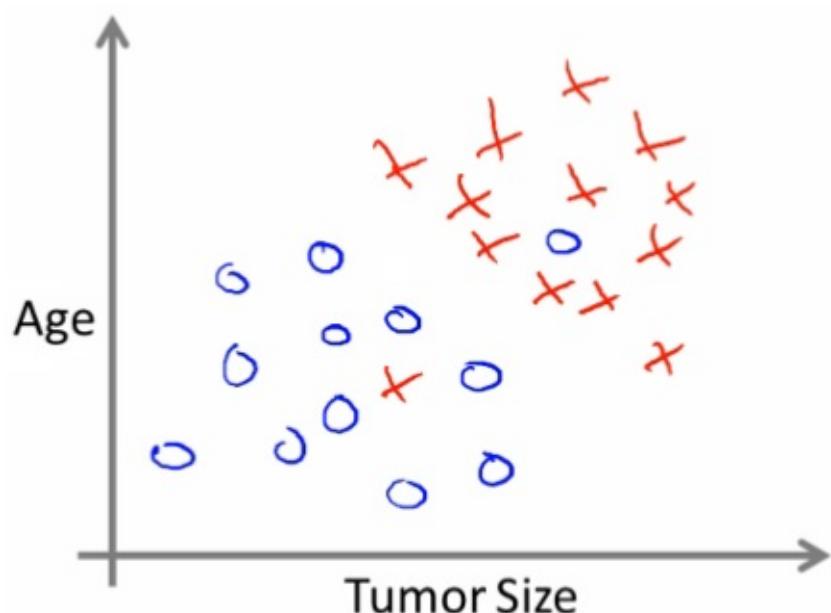
ডেটাসেট থেকে যেসব তথ্য পাওয়া যায়

- এটা একটা Classification প্রবলেম, কেননা আমরা ইনপুটে দিচ্ছি Tumor Size এবং আউটপুটে Yes/No টাইপ উভয় চাচ্ছি। কোন মান চাচ্ছি না, যেমন বাড়ির দামের ক্ষেত্রে আমরা একটা মান চাচ্ছিলাম, Yes/No টাইপ উভয় গ্রহণযোগ্য ছিল না। তাই ওটা বিশেষজ্ঞ প্রবলেম
- ডেটাকে দুইভাগে ভাগ করতে হবে, মাঝখানে কিছুই থাকবে না, 1 কে Malignant এবং 0 কে Not Malignant হিসেবে আমরা ট্যাগ দিতে পারি।
- এটা যদি ভিন্নভাবে ফট করা যায়



- এখনে আমরা একটা মাত্র প্যারামিটার (Size of Tumor) ব্যবহার করে বলার চেষ্টা করছি সেটা প্রাণঘাতী কি না? আসলে একধরণের ইনপুট থাকবে এমন কোন কথা নাই, আরও অনেক ইনপুট থাকতে পারে। যেমন Age vs Tumor size

ইনপুট প্যারামিটার যদি আরও থাকে



সমাধানের উপায়?

আমরা একটা স্ট্রেট লাইন দিয়ে এই গ্রুপ দুইটাকে আলাদা করতে পারি। যেটা নিয়ে পরে আলোচনা করা হবে।

পৰিবৰ্তীতে আমৰা Unsupervised Learning এৰ পাশাপাশি প্ৰয়োজনীয় পাইথন সফটওয়্যাৰ প্যাকেজ ইন্টলেশন দেখব।

মেশিন লার্নিং পাইথন প্যাকেজ ইন্টলেশন

মেশিন লার্নিংয়ের জন্য কিছু পাইথন মডিউল ও লাইব্রেরি প্রয়োজন। আমরা মডেল স্র্যাচ থেকে বিন্দু করার পশ্চাপাশি দেখব কীভাবে লাইব্রেরি ব্যবহারের মাধ্যমেও মডেল তৈরি করা যায়।

Anaconda প্যাকেজ ডাউনলোড ইন্টলেশন (পাইথন ২.৭)

সম্পূর্ণকোর্সে আমরা পাইথন ২.৭ ভার্সনটি ব্যবহার করব। তাই অ্যানাকোন্ডা প্যাকেজের পাইথন ভার্সনও ২.৭ হওয়া বাঞ্ছনীয়।

উইন্ডোজ

- [উইন্ডোজ ৩২ বিটের জন্য ডাউনলোড \(335MB\)](#)
- [উইন্ডোজ ৬৪ বিটের জন্য ডাউনলোড \(281MB\)](#)

ডাউনলোড করে সাধারণ সফটওয়্যার ইন্টল করেন যেভাবে সেভাবে ইন্টল করলেই হবে। Start Menu তে গিয়ে Spyder সার্চ দিলেই IDE টি পেয়ে যাবেন।

OSX

- [৬৪ বিট ডাউনলোড](#)

লিনাক্স

- [৬৪ বিট ডাউনলোড \(392MB\)](#)
- [৩২ বিট ডাউনলোড \(332MB\)](#)

ডাউনলোড শেষে ডাউনলোড ডিরেক্টরিতে গিয়ে নিচের কমান্ড দিন টার্মিনালে,

```
bash Anaconda2-4.0.0-Linux-x86_64.sh
```

Spyder IDE

Spyder IDE ওপেন করুন ও নিচের কোডটি রান করুন, যদি কাজ করে তাহলে বুঝতে হবে আপনার কম্পিউটার মেশিন লার্নিংয়ের জন্য প্রস্তুত।

```
from sklearn.linear_model import LinearRegression  
print sklearn.__version__
```

The screenshot shows the Spyder IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. Below the menu is a toolbar with various icons. The left sidebar is the Project explorer, showing a file named 'PyQtPlotter' and 'py_plotter.py'. The main workspace has tabs for 'IPython console' and 'Console 1/A'. The IPython console tab is active, displaying the following Python code:

```
target = [
    [0, 0],
    [70, 70],
    [70, 70],
    [0, 64],
    [0, 68],
    [0, 72],
    [0, 72],
    [64, 0],
    [68, 0],
    [72, 0],
    [72, 0]
]
hidden1 = 8
hidden2 = 2
output = 2
```

The code defines a list of target coordinates and sets hidden layer sizes to 8, 2, and an output layer size of 2. The bottom status bar shows permissions as RW, end-of-lines as CRLF, encoding as UTF-8, line 34, column 1, and memory usage at 88%.

Anaconda Official Website

অফিশিয়াল ওয়েবসাইট

পরবর্তী পর্বে আমরা রিপ্রেশন অ্যানালাইসিস দেখব উদাহরণের মাধ্যমে।

একনজরে পাইথন মেশিন লার্নিং টুলস

- Pandas
 - ডেটা ফ্রেম লাইব্রেরি
- Numpy
 - ক্যালকুলেশন ও ম্যাট্রিক্স লাইব্রেরি (লিনিয়ার অ্যালজেব্রা)
- Scikit-learn
 - মেশিন লার্নিং লাইব্রেরি
- IPython/Jupyter Notebook
 - মেশিন লার্নিং প্রোগ্রাম সহজ করে লেখার টুল

There are only two kinds of languages: the ones people complain about and the ones nobody uses -- Bjarne Stroustrup

মেশিন লার্নিংয়ের জন্য পাইথন লাইব্রেরি

মেশিন লার্নিংয়ের জন্য পাইথনের যেসব লাইব্রেরি ব্যবহার করা হবে:

- `numpy` - সায়েন্টিফিক ক্যালকুলেশনের জন্য
- `pandas` - ডেটা ফ্রেম
- `matplotlib` - টি ও ত্রিমাত্রিক গ্রাফ প্লটিংয়ের জন্য
- `scikit-learn`
 - মেশিন লার্নিং অ্যালগরিদম
 - ডেটা প্রি প্রসেসিং
 - প্রেডিক্টিভ মডেল বিল্ডিং ও পারফর্মেন্স টেস্টিং
 - ... আরও অনেক কিছু
- IPython Notebook / Jupyter Notebook
 - Painless Machine Learning মডেল তৈরির জন্য

IPython Notebook / Jupyter Notebook

Jupyter Notebook আগে IPython Notebook হিসেবে পরিচিত ছিল।

কেন IPython Notebook সম্পর্কে জানা প্রয়োজন?

- একটা নোটবুক আমরা যেসব কাজে ব্যবহার করে থাকি। IPython Notebook কে প্রোগ্রামারের নোটবুক বললে ভুল বলা হবে না।
- মেশিন লার্নিংয়ের কাজগুলো যেহেতু ইটারেবল, মানে কাজ করার পাশাপাশি প্রায়ই কাজের আগের অংশ ও পরের অংশ চেক করতে হয় সেজন্য IPython Notebook মেশিন লার্নিংয়ের জন্য পার্ফেক্ট টুল।
- কোড শেয়ারিংয়ের ক্ষেত্রে আমরা কোড শেয়ার করি কিন্তু যার সাথে শেয়ার করা হয় তাকে নিশ্চয়ই কোড রান করে দেখতে হয়। IPython Notebook এর ক্ষেত্রে ডকুমেন্টগুলো শেয়ারেবল। প্রতিটি কমান্ডের বা কমান্ড বাস্তুলের আউটপুট একটি ডকুমেন্টের মাধ্যমে শেয়ার করা সম্ভব।
- আরেকটি বড় সুবিধা হল IPython Notebook পুরোপুরি Markdown ফর্ম্যাটিং সাপোর্টেড। ইচ্ছা করলে আপনি নোট আকারে কথাবার্তা Markdown Format এ লিখে দিতে পারেন।
- IPython Notebook পাইথনের পাশাপাশি: `C#, Scala, PHP ..` ইত্যাদি অন্যান্য ল্যাঙ্গুেজও সাপোর্ট করে, তবে সেক্ষেত্রে প্লাগিন ব্যবহার করতে হবে।

IPython Notebook বান করা

cmd ওপেন করে লিখুন `ipython notebook` তারপর Enter চাপুন। এতে কাজ না করলে লিখুন `jupyter notebook`।

দুইটার একটা কাজ করবেই, কাজ না করলে Anaconda প্যাকেজ রিস্ট্রিস্টেশন দিন।

নোটবুক ডেমো

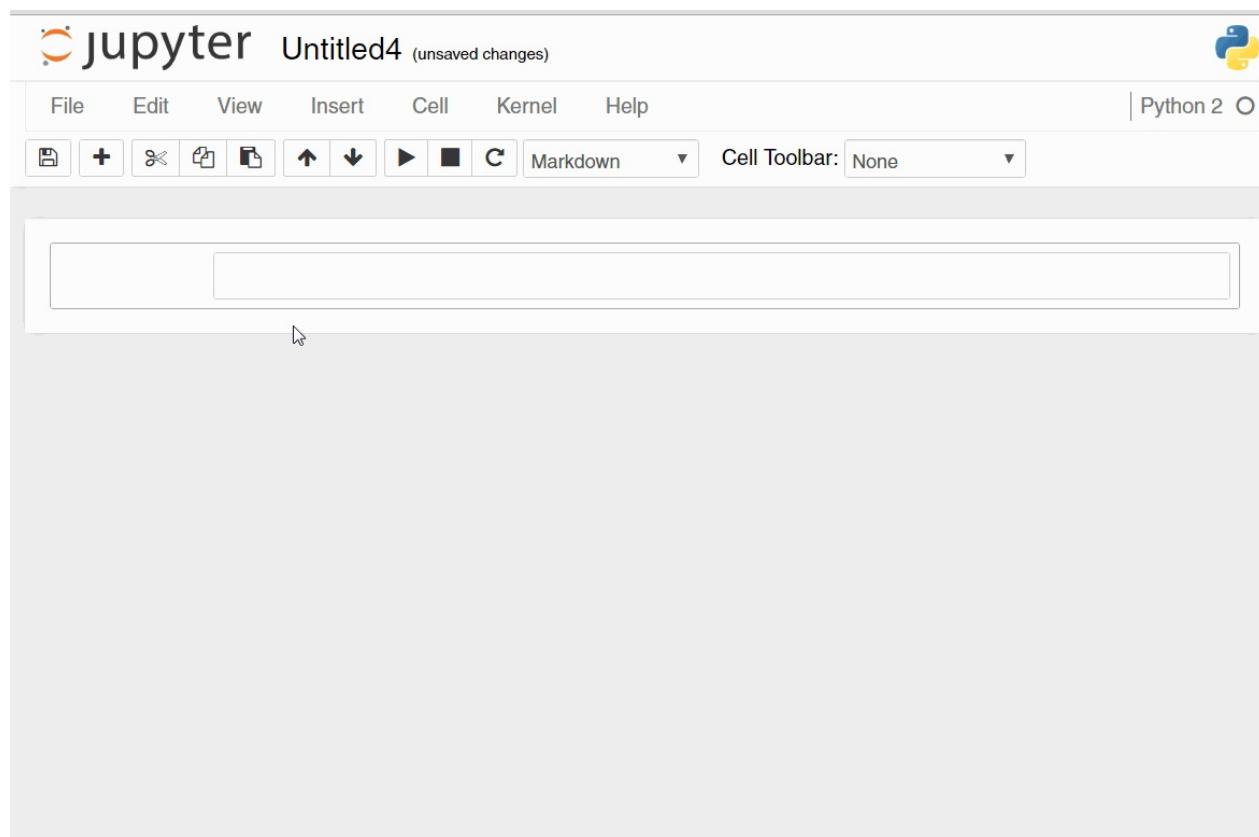
একটা ছোট ডেমো

বেসিক ইলেক্ট্রোক্ষন

- কোড লিখে Enter চাপলে নতুন লাইনে লেখা যাবে
- Shift + Enter চাপলে একটা Cell এক্সিকিউট হবে



নোট ও কোড একসাথে (বাংলা সাপোর্টড!)



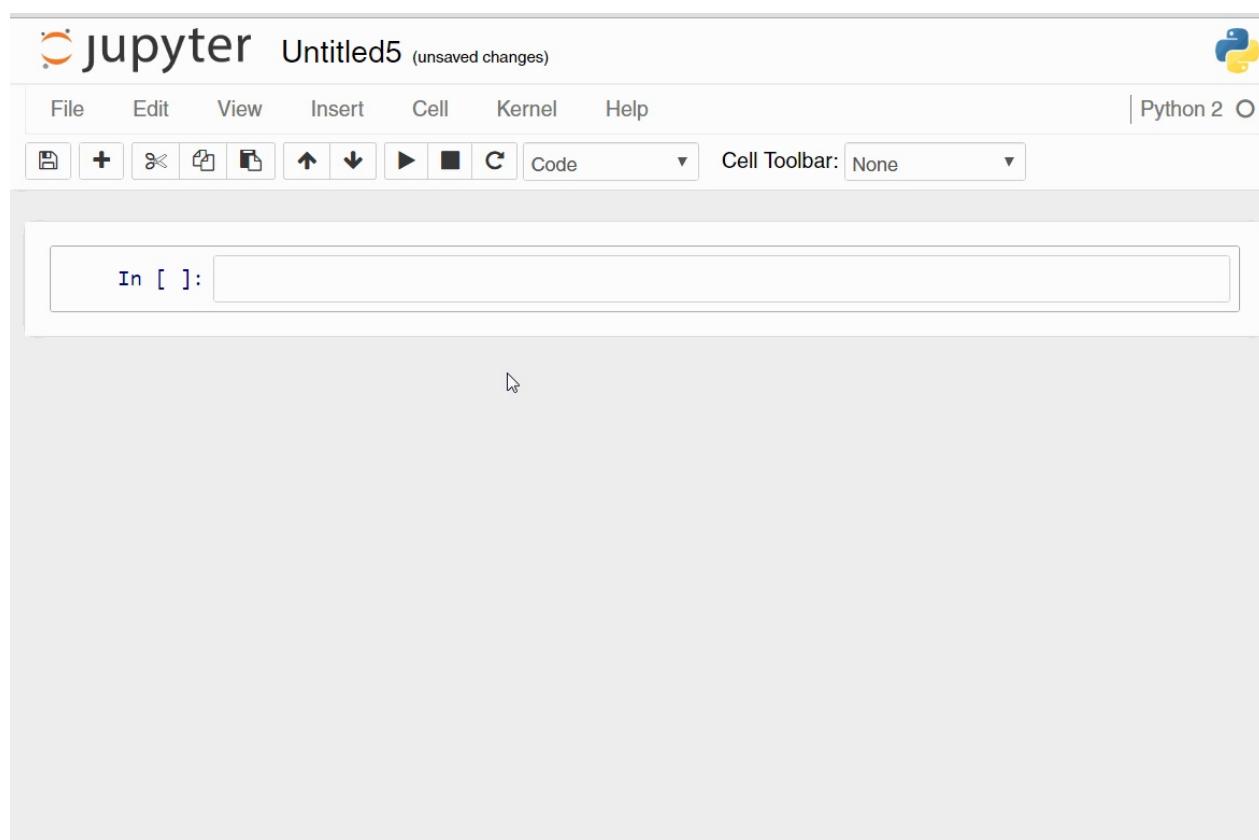
আরও একটুখানি IPython Notebook!

ইনলাইন গ্রাফ প্রটিঃ!

```
%matplotlib inline
from matplotlib import pyplot as plt
import numpy as np

x = np.array(range(10))
y = np.array(range(10))

plt.plot(x, y)
plt.show()
```

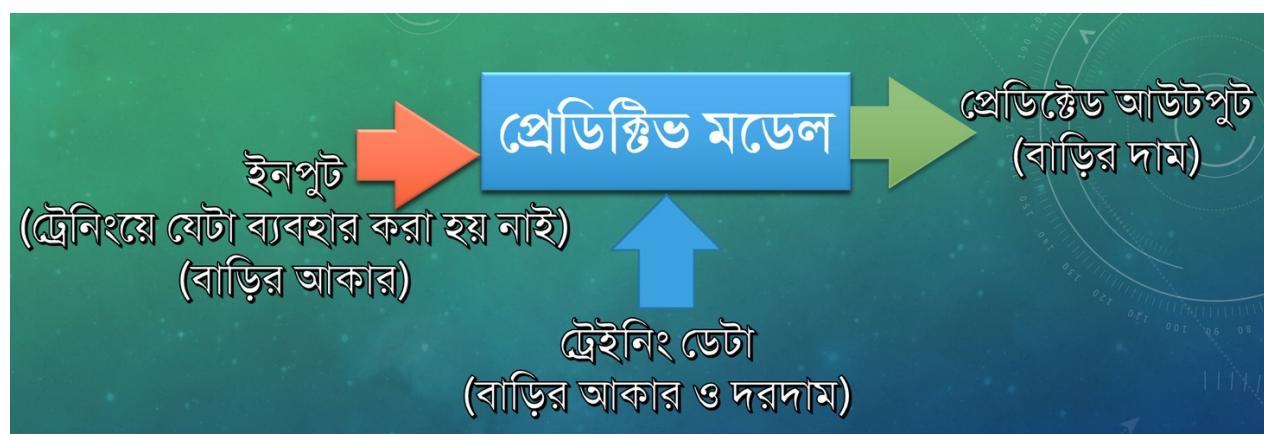


IPython এর কাজ দেখানো এই প্রক্ষেত্রে! পরবর্তীতে নতুন প্যাকেজগুলোর সাথে পরিচয় করিয়ে দেওয়া হবে।

মেশিন লার্নিংয়ের কাজের ধারা

মেশিন লার্নিং অ্যাপ্লাই করার আগে বেশ কিছু জিনিস সম্পর্কে আগে জানতে হয়। আমরা এই চ্যাপ্টারে দেখব, মেশিন লার্নিং অ্যালগরিদম চয়েস থেকে শুরু করে কীভাবে প্রেডিষ্টিভ মডেল বিন্দ করবেন সেটা একটা সাধারণ বেসিপির মাধ্যমে প্রকাশ করা যায়।

scikit-learn এর তিনটি অংশ:



মডেল

scikit-learn লাইব্রেরি ব্যবহার করার সময় সবচেয়ে সাধারণ কাজটি হল যে ল্লাসিফায়ার ব্যবহার করবেন তার একটা অবজেক্ট (মডেল) তৈরি করা। যেমন, আগের বাড়ির আকার ও দরদাম সম্পর্কিত সমস্যার ক্ষেত্রে,

```

from sklearn import LinearRegression
#Creating the regression model
linear_regression_model = LinearRegression()

```

আমাদের কাজ হবে এই লিনিয়ার রিগ্রেশন মডেলকে ডেটা দিয়ে ট্রেইন করা।

ফিট (ট্রেইন)

```

#Let, house_sizes = Contains all size of house
#house_prices = Contains all price of house

#Training the model
linear_regression_model.fit(house_sizes, house_prices)

```

সাধারণত প্রত্যেকটা মডেল এ `fit` ও `predict` এই দুইটা ফাংশন প্রায়ই থাকে।

প্রেডিক্ষন

```
#predicted value
"""
Here, test_house_size is a variable which is not a member of training data, rather a unique one
"""
predicted_price = linear_regression_model.predict(test_house_size)
```

এই `predicted_price` ড্যারিয়েবলে আমি যে সাইজের বাড়ির দাম জানতে চাচ্ছি সেটা অ্যাসাইন হবে।

মেশিন লার্নিং ওয়ার্কফ্লো বলতে কী বোঝায়?

কেতাবি সংজ্ঞানুসারে,

An orchestrated and repeatable pattern which systematically transforms and processes information to create prediction solutions.

An orchestrated and repeatable pattern:

এর মানে, একই ওয়ার্কফ্লো দিয়ে আমরা সমস্যা ডিফাইন করব, এবং সেই ওয়ার্কফ্লো দিয়েই আমরা সল্যুশন বিন্দ করব।

Transforms and processes information:

ডেটা দিয়ে মডেল তৈরি করার আগে সেটাকে ট্রেনিংয়ের জন্য ব্যবহারযোগ্য করে নিতে হবে।

ধৰা যাক আমরা একটা প্রেডিক্ষিত মডেল তৈরি করতে চাচ্ছি যেটা হাঁ বা না তে উত্তর দেয়। ইনপুট ডেটা যদি নিউমেরিক্যাল হয় তাহলে আউটপুটও নিউমেরিক্যাল হলে সুবিধা। এই কারণে, ট্রেনিং ডেটার ক্ষেত্রে যত হাঁ/না ওয়ালা লেবেল আছে ওগুলো আমরা রিপ্লেস করে 1 ও 0 বসিয়ে দিতে পারি। এটাকে ইনফর্মেশন প্রিপ্রেসিং বলে।

Create prediction solutions:

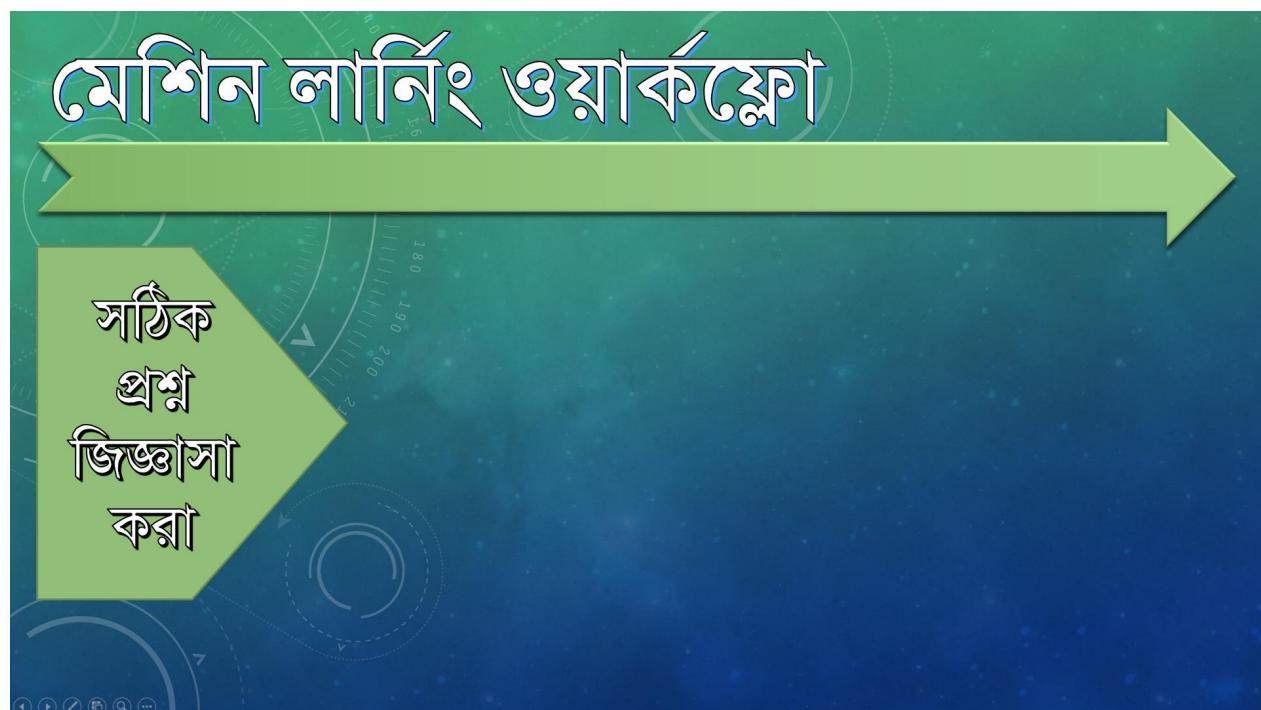
যেকোন মেশিন লার্নিংয়ের সর্বশেষ লক্ষ থাকে প্রেডিক্ষন করা। তবে প্রেডিকশন যেন গ্রাহকের চাহিদা মেটায় সেক্ষেত্রেও নজর রাখতে হবে।

যেমন, আমার তৈরি একটা মডেলের নতুন ডেটা ট্রেনিংয়ে লাগে ২ দিন সময়, প্রেডিক্ষন করতে লাগে আরও ১ দিন সময়। এখন গু ১ দিনে আরও নতুন ডেটা যদি আসে, সেগুলো ট্রেইন করতে আমার আরও সময় প্রয়োজন। ততক্ষণে ডেটা প্রেডিক্ষন করার টাইম লিমিট আরও বেড়ে যাবে। এই মডেল কী কোন সুস্থ স্বাভাবিক মানুষ গ্রহণ করবে? অবশ্যই

না, তাই যত কম ট্রেনিং সময়ে একটা মডেল যত কাছাকাছি উত্তর প্রেডিষ্ট করতে সক্ষম হয় সেই অ্যালগরিদম ও মেশিন লার্নিং সিস্টেম তত ভাল।

মেশিন লার্নিং ওয়ার্কফ্লো:

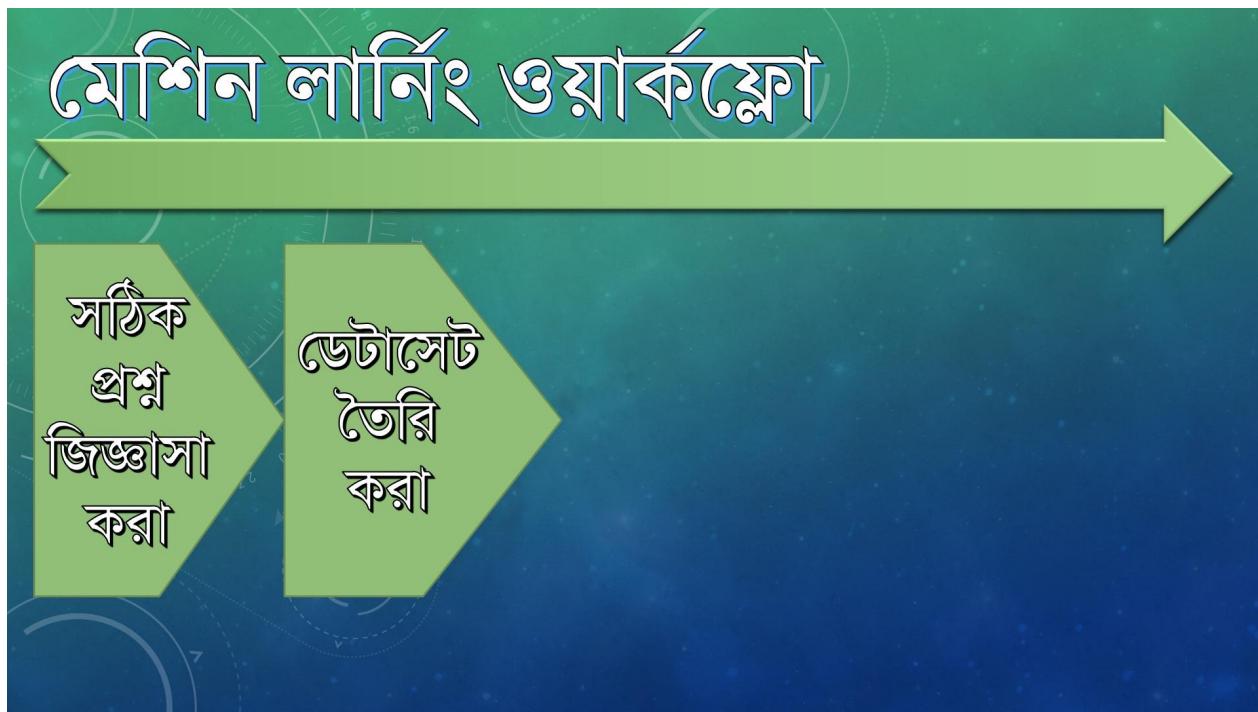
সঠিক প্রশ্ন জিজ্ঞাসা করা



আমি আসলে কী করতে চাই, এটা দিয়ে শুরু হয় কাজ। বাড়ির আকার ও দামের সমস্যার ক্ষেত্রে বোধ্য যায় আমি চাচ্ছি বাড়ির আকার কে ইনপুট দিতে এবং আউটপুটে চাচ্ছি সেটার দাম।

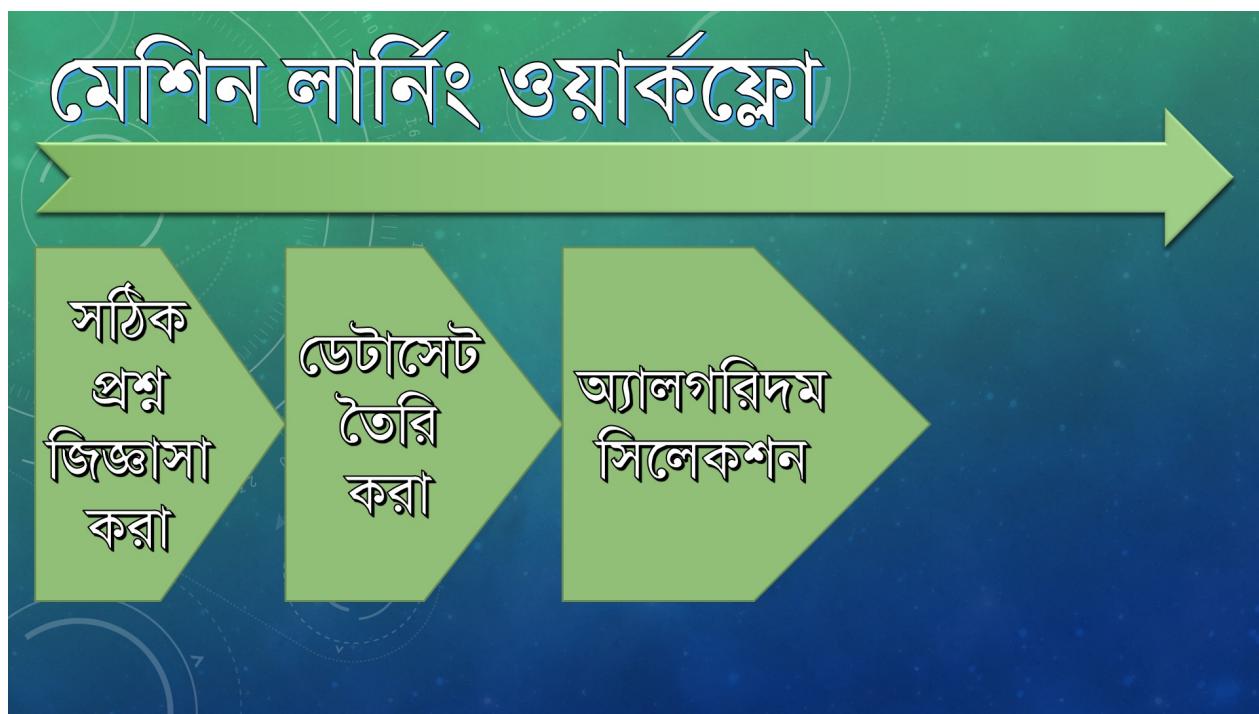
জটিল কাজ সহজ হতে পারে যদি সঠিক প্রশ্ন সেট করার পর আমরা তার উত্তর খুঁজি।

ডেটা গুচ্ছানো



এবাব আমার সমস্যা সমাধানের জন্য বা মডেল ট্রেইন করার জন্য অবশ্যই ডেটা চিনিয়ে দিতে হবে। একটা মেশিন তখনই একটা ভাল কাজ ও খারাপ কাজের মধ্যে পার্থক্য বের করতে পারবে যদি তাকে ভাল ও খারাপ কাজ দেখিয়ে ট্রেইন করা হয় তাহলেই পরে সে নতুন কোন ঘটনা দেখে তার ট্রেনিং ডেটা থেকে মিলিয়ে নিতে পারবে সেটা ভাল না খারাপ কাজ।

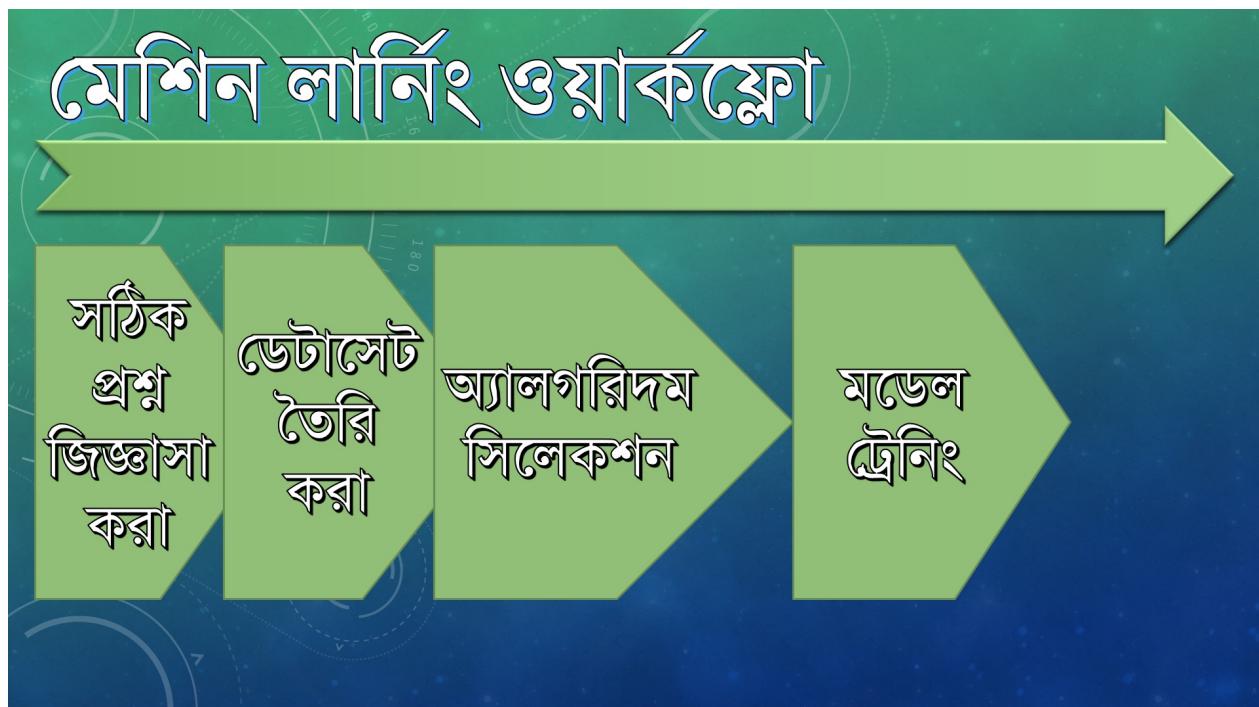
অ্যালগরিদম সিলেকশন করা



যেহেতু একই কাজ বিভিন্ন মডেল দিয়ে করা যায় তাই যে মডেল এর কম দেখাবে সেটাই সাধারণত সিলেক্ট করা হয়।

অ্যালগরিদম সিলেক্ট করার জন্য অবশ্যই প্রবলেমসেট ও অ্যাডেইল্যাবল ডেটাসেট বুঝতে হবে। যেমন বাড়ি ও দরদামের সমস্যাটি একটি রিগ্রেশন প্রবলেম, আমি যদি এখানে ক্লাস্টারিং (ডেটা ক্লাসিফিকেশন) অ্যালগরিদম অ্যাপ্লাই করি তাহলে অবশ্যই এর প্রেডিকশন খুবই বাজে আসবে।

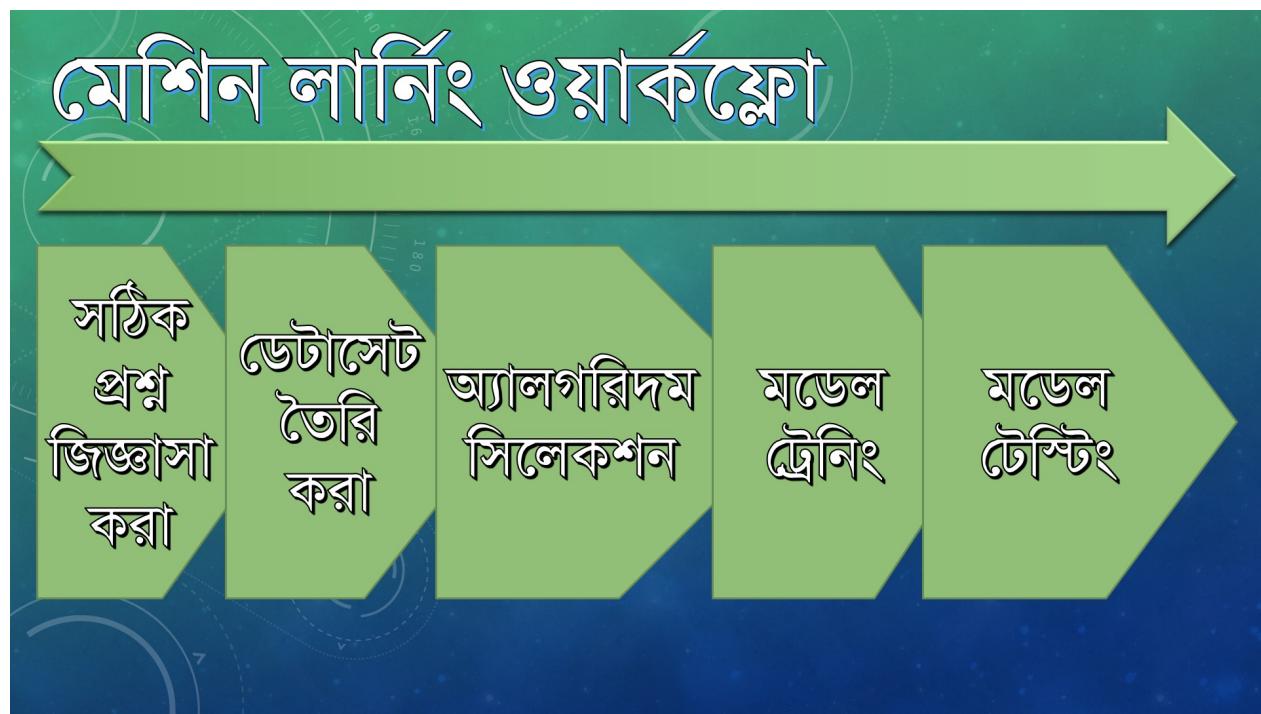
মডেল ট্রেইনিং



আমার হাতে থাকা ডেটাসেট দুইভাগে ভাগ করব, একটা হল Training Dataset এবং আরেকটি হল Testing Dataset।

Testing Dataset এর কোন ডেটা-ই Training Dataset এ থাকে না। যদি থাকত তাহলে মেশিন লার্নিংয়ের মূল উদ্দেশ্যই মাটি হয়ে যেত। (মানে তৈরিকৃত মডেল নতুন কিছু শেখার পরিবর্তে ডেটা মুখ্যত করা শিখত, যেমনটা আমরা নিজেরা পরীক্ষা দেওয়ার সময় করে থাকি, তাই না?) এই ট্রেইনিং ডেটা মডেলে Feed করে ও লার্নিং অ্যালগরিদম অ্যাপ্লাই করার মাধ্যমে মডেল ট্রেইনিং সম্পন্ন করা হয়।

মডেল টেস্টিং



Dataset দুইভাগে ভাগ করার কারণ হচ্ছ, ট্রেইনিং ডেটা দেওয়ার মাধ্যমে মডেলকে আগে শিখাতে হবে। তারপর একটা ইনপুট চয়েস করতে হবে (Testing Dataset) ও ইনপুট টি মডেলকে দিলে তার প্রেডিক্ষেন আউটপুট করতা সঠিক বা ভুল সেট বের করা যাবে।

তারমানে আমার হাতের সব ডেটাই আমি ট্রেনিংয়ে দিচ্ছি না, তারমানে কিছু ডেটা আমার হাতে থেকে যাচ্ছে যেটা দিয়ে আমি ডেরিফাই করতে পারব আবু আমার তৈরি করা মডেলটি কিছু শিখতে পারছে কিনা।

ব্যাপারটা অনেকটা এরকম, ধরি আমার তৈরি করা ML মডেলকে আমি ৩ এর ঘরের নামতা শেখাতে চাই। তাহলে আমি এরকম একটি Dataset তৈরি করব,

$$\begin{aligned}
 1 \times 1 &= 1 \\
 1 \times 2 &= 2 \\
 1 \times 3 &= 3 \\
 1 \times 4 &= 4 \\
 1 \times 5 &= 5 \\
 1 \times 6 &= 6 \\
 1 \times 7 &= 7 \\
 1 \times 8 &= 8
 \end{aligned}$$

এবার আমি যদি এখান থেকে ট্রেনিং ডেটা ও টেস্টিং ডেটা আলাদা করি তাহলে ডেটাসেটগুলো দাঁড়াবে এরকম,

দ্রষ্টব্য: ডেটাসেট থেকে ট্রেনিং ডেটা ও টেস্টিং ডেটা আলাদা করার জন্যও আলাদা অ্যালগরিদম আছে। ডেটাসেট যদি অনেক কম হয় সেসব ক্ষেত্রে ডেটা সিলেকশনের সফিস্টিকেটেড অ্যালগরিদমগুলো খুবই ভাল কাজ করে। আমরা পরে বিস্তারিত দেখব।

ট্রেনিং ডেটা

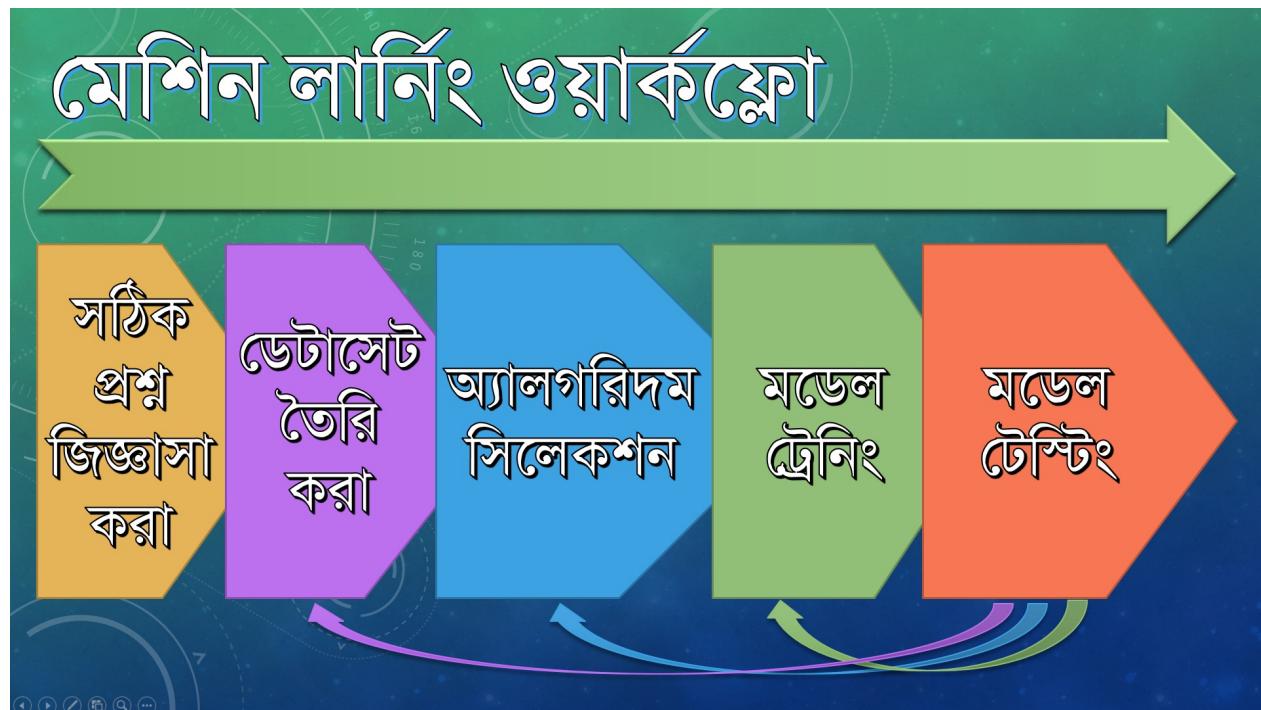
$1 \times 1 = 1$
 $1 \times 2 = 2$
 $1 \times 3 = 3$
 $1 \times 4 = 4$
 $1 \times 5 = 5$
 $1 \times 6 = 6$
 $1 \times 7 = 7$
 $1 \times 8 = 8$

টেস্টিং ডেটা

$1 \times 9 = 9$

এখনে পরিষ্কার দেখা যাচ্ছে আমি ট্রেইনিং ডেটায় 3×5 দেই নাই। তার মানে আমি 3×5 বাদে বাকি ডেটা দিয়ে মডেল ট্রেইন করব। তারপর 3 ও 5 ইনপুট দিয়ে দেখব মডেলটি আউটপুট 15 এর কাছাকাছি দিচ্ছে কিনা। যদি দেয় তার মানে আমার মডেলের অ্যালগরিদম সিলেকশন, ডেটা প্রিপ্রসেসিং ও ট্রেইনিং যথাযথ হয়েছে। আর যদি এটা কাজ না করে সেক্ষেত্রে আমার আবার ডেটা প্রিপ্রসেসিং থেকে শুরু করে নতুন অ্যালগরিদম দিয়ে ট্রাই করতে হবে।

মডেল তৈরি করা পুরোটাই ইটারেচিভ প্রসেস, মানে বারংবার করার মাধ্যমেই পারফেকশন আনতে হয়, একবারেই একটা মডেল পারফেক্ট হবে এমন ধারণা করা ঠিক নয়।



ওয়ার্কফ্লো গাইডলাইন

শেষের স্টেপ না, মডেল বিল্ড করার জন্য সতর্ক থাকতে হবে শুরু থেকেই

মডেল বিন্দু করার জন্য শুরু থেকেই সব চিন্তাভাবনা করে আগামে হবে, কেননা প্রতিটি স্টেপ ই পূর্ববর্তী স্টেপের উপর নির্ভরশীল। অনেকটা চেইন এর মত, একটা অংশ ডুল করলেই আবার প্রথম থেকে শুরু করতে হবে।

তারমানে আল্টিমেট টাগেট, হাতে থাকা ডেটাসেট, অ্যালগরিদম সিলেকশন সবকিছুই করতে হবে যন্ত্র ও চিন্তাভাবনার সাথে।

যেকোন সময় আগের স্টেপে ফিরে যেতে হতে পারে

ধরুন, আপনার কাছে গুণের ডেটাসেট আছে কিন্তু আপনি আউটপুটে চাচ্ছেন দুইটি সংখ্যার যোগের ফলাফল। তারমানে আপনি যা চান তার সাথে ডেটাসেট এর কোন মিল নাই। তাই আমাদের এবার গুণের ডেটাসেট কে রিপ্লেস করতে হবে যোগের ডেটাসেট দিয়ে তারপর আবার মডেলকে ট্রেইন করতে হবে।

ডেটা সাজানো লাগবেই

RAW ডেটা কখনোই আপনার মনমত সাজানো থাকবে না, মডেল ট্রেইনিংয়ের জন্য সেটাকে অবশ্যই প্রিপ্রেস করতে হবে।

ডেটা প্রিপ্রেসিং এই মূলত সবচেয়ে বেশি সময় লাগে।

ডেটা যত মজা তত

আসলে আপনি মডেলে যত ডেটা ফিল্ড করতে পারবেন তার প্রেডিকশন অ্যাকুরেসি ততটাই বেটার হবে। এই খিওরি স্বত্ত্বসিদ্ধ।

প্রবলেম হোক এটা সেটা সল্যুশন হোক ভাল

কখনোই খারাপ সল্যুশনকে পাতা দেবেন না। একটা প্রবলেম সল্ভ করার সময় অনেক চেষ্টার পরেও যদি আশানুরূপ পার্ফর্মেন্স না পান, সেক্ষেত্রে অন্যান্য স্টেপগুলো সম্পর্কে প্রশ্ন করুন।

- আমি কি সঠিক প্রশ্ন করছি?
- সমস্যা সমাধান করার মত প্রয়োজনীয় ডেটা কি আমার আছে?
- আমি সিলেক্ট করা অ্যালগরিদম কি সঠিক?

যদি সন্তোষজনক উত্তর না পান তাহলে সমস্যার সমাধান না করাই বেটার। কারণ যে মডেল ৫০% সময় সঠিক উত্তর দেয় আর বাকি ৫০% ডুল উত্তর দেয় সেটা কনফিউশন তৈরি করার জন্যই ভাল, সমস্যা সমাধানের জন্য নয়।

আপাতত এই পর্যন্তই। পরবর্তী পর্বে আমরা মেশিন লার্নিংয়ের ১ম ধাপ দেখব। "কীভাবে সঠিক প্রশ্ন করতে হয়?"

কীভাবে সঠিক প্রশ্ন করতে হয়?

"There are no right answers to wrong questions." - Ursula K. Le Guin

"Ask the right questions if you're going to find the right answers." - Vanessa Redgrave

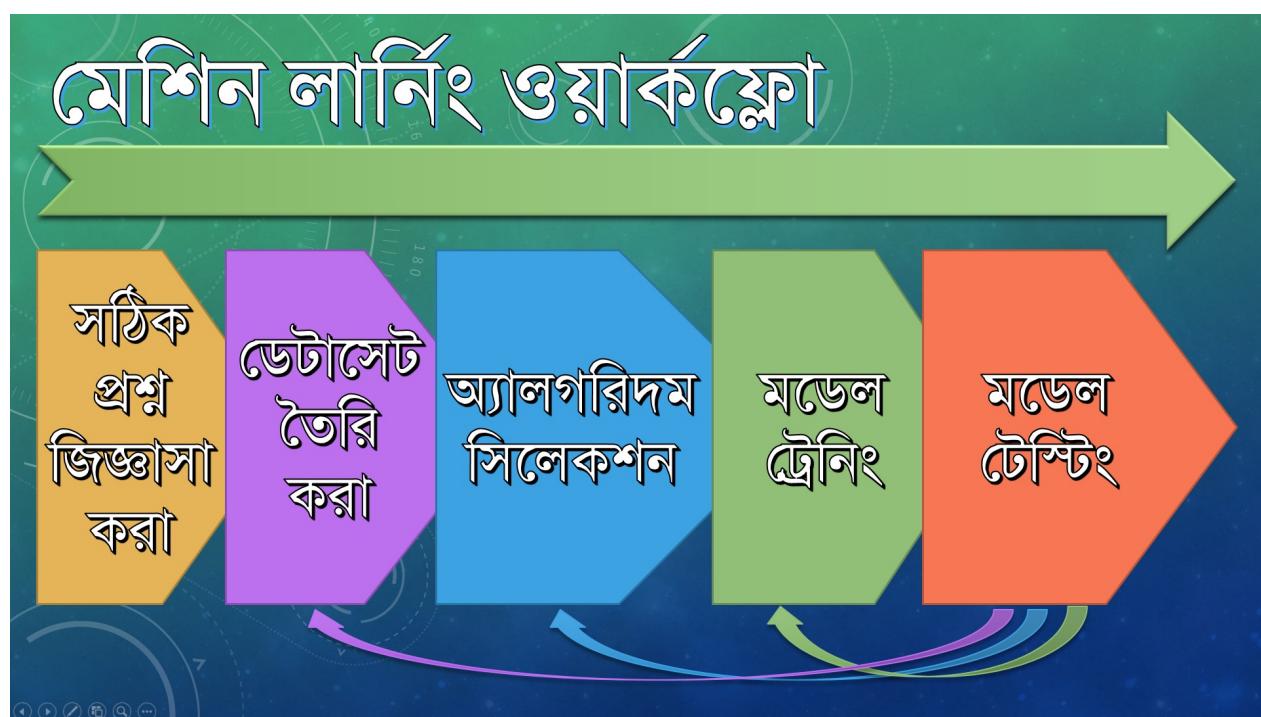
"In school, we're rewarded for having the answer, not for asking a good question." - Richard Saul Wurman

মেশিন লার্নিং মডেল তৈরির প্রথম স্টেপ

সঠিক প্রশ্ন করবেন কীভাবে?

ওয়ার্কফ্লো রিডিশন

যেহেতু আজকের টপিক 'সঠিক প্রশ্ন জিজ্ঞাসা করা' বা 'সঠিক প্রশ্ন তৈরি করা' সেহেতু গত পর্বের ওয়ার্কফ্লো একবার দেখে নেওয়া যাক।



সমস্যা

আমরা যদি জেনারেল মেথড অ্যাপ্লাই করতে চাই তাহলে সর্বপ্রথম দরকার একটা সমস্যাকে সেই মেথড গুলো দিয়ে সমাধান করা। আমাদের বাছাইকৃত সমস্যাটি একবার দেখা যাক,

সমস্যার বিবরণ

একজন লোক ডায়াবেটিস এ আক্রান্ত হবে কিনা প্রেডিষ্ট করুন

কাজের শুরুতেই কি সঠিক প্রশ্ন পেয়ে গেলাম না?

সমস্যার বিবরণ দেখে মনে হতে পারে, আমরা তো আমাদের কাঞ্চিত প্রশ্ন পেয়েই গেলাম, নতুন করে প্রশ্ন করার মানে কী? এই সমস্যাটা সল্ভ করলেই তো হচ্ছে?!

উত্তর হল, না। মেশিন লার্নিং সমস্যাকে শুধু এক লাইনের একটা প্রশ্ন দিয়ে ডিফাইন করা ভুল, এই প্রশ্নকে ভেঙ্গে ক্ষুদ্র ক্ষুদ্র ও নির্দিষ্ট কিছু প্রশ্নে ভাগ করতে হবে, তারপর সেগুলো সমাধান করতে হবে।

তাহলে কীরকম হবে প্রশ্ন গুলো?

- প্রশ্নগুলো এমন হতে হবে যেগুলোর সমাধান করলে আসলেই আমরা একটা Fully Functional Predictive Model তৈরি করতে পারি।
- প্রশ্নের উপরে ভিত্তি করে আমরা প্রেডিষ্টিভ মডেল বিন্দু করছি, তাই আমাদের To the point প্রশ্ন দরকার। যার ক্রাইটেরিয়ার উপর ভিত্তি করে আমরা সল্যুশন তৈরি করব।
- একটা এক লাইন প্রশ্নের থেকে আরও দরকারী হচ্ছে প্রশ্নের মধ্যে থাকা কিছু স্টেটমেন্ট, যেগুলো আমাদের সল্যুশন বিন্দু এর শুরু ডিফাইন করে, শেষ ডিফাইন করে (যেমন, মডেল এর প্রেডিকশন সাক্ষেত্রে বেট কর্তৃ ভাল হলে আমরা মডেল অপ্টিমাইজেশন বাদ দিয়ে প্রেডিষ্ট করতে বসে যাব) এবং কীভাবে আমরা আমাদের লক্ষ্য পৌঁছাব।

তাহলে Solution Statement Goal গুলো দেখা যাক

- ডেটা সোর্স পর্যবেক্ষণ (Scope & Dataset)
- পার্ফর্মেন্স স্কোর ও পার্ফর্মেন্স টার্গেট
- যেখানে ব্যবহার করা হবে (Context)
- কীভাবে সল্যুশন তৈরি করা হবে

এই পয়েন্টগুলো একটার পর একটা যুক্ত করলেই আমরা আমাদের কাঞ্চিত প্রশ্নগুলো পেয়ে যাব।

স্কোপ ও ডেটা সোর্স:

অ্যামেরিকান ডায়াবেটিস ওয়েবসাইট ডেটাসেট পর্যবেক্ষণ করে বেশ কিছু ডায়াবেটিসে আক্রান্ত হবার বেশ কিছু ফ্যাট্টের তুলে ধরেছে যেটা আমাদের ডেটাসেট এর শুরুপূর্ণ ইনপুট ড্যারিয়েবল চিহ্নিত করতে সাহায্য করবে।

বয়স (Age)

বয়স্কদের ডায়াবেটিসে আক্রান্ত হওয়ার সম্ভাবনা বেশি থাকে।

জাতি (Race)

African-American, Asian-American, American-Indian দের ডায়াবেটিস হ্বার সম্ভাবনা বেশি।

লিঙ্গ (Gender)

ডায়াবেটিসে আক্রান্ত হওয়ার জন্য লিঙ্গের কোন প্রভাব নেই।

উপর্যুক্ত Factor গুলো কেন গুরুত্বপূর্ণ?

- ইনপুট ড্যায়িয়েবল ফিল্টারিংয়ে: ফ্যাট্টারগুলো থেকে দেখা যায় যে, আমরা 'Race' এর উপর গুরুত্ব দেব বেশি এবং 'Gender' এর ক্ষেত্রে গুরুত্ব না দিলেও হবে। যদি আমরা Gender কে ইনপুট ড্যায়িয়েবল হিসেবে নেই তাহলে প্রেডিকশন অবশ্যই খারাপ আসবে।
- ডেটাসেট বাছতে: যেসব ডেটাসেটে Age, Race (আরও থাকতে পারে) আছে, সেসব ডেটাসেট আমাদের বাছাই করতে হবে।
 - আমরা এই কাজের জন্য University of California Irvine (UCI) এর রিপোজিটরি থেকে [Pima Indian Diabetes Study](#) সিলেক্ট করব। কারণ এই ডেটাসেট আমাদের ডিমান্ড করা ক্রাইটেরিয়াগুলো ফিলাপ করে।

পরিবর্তিত স্টেটমেন্ট

স্কোপ ও ডেটাসেট পর্যবেক্ষণের পর আমাদের পরিবর্তিত প্রবলেম স্টেটমেন্ট

Pima Indian Diabetes ডেটাসেট ব্যবহার করে বের করতে হবে কে কে ডায়াবেটিসে আক্রান্ত হবে

পার্ফর্মেন্স স্কোর ও পারফর্মেন্স টার্গেট

- সমস্যার সমাধান যতই জটিল হোক, আউটপুট আমরা সহজেই ধারণা করতে পারছি, এই সমস্যার সমাধান হ্যাঁ/না আকারে আসবে। অর্থাৎ Binary Result (True or False)
- মডেল বিন্দ করলে তার একটা পার্ফর্মেন্স স্কোর আমরা পাব। মানে আমাদের মডেল কতটা ভাল প্রেডিক্ষন করতে পারে। কিন্তু এই পার্ফর্মেন্সের একটা লিমিট আছে। সাধারণত 100% প্রেডিকশন রেট হ্যাঁ না, কিন্তু আমাদের চেষ্টা থাকতে হবে কতটা ভাল করা যায়।
- তাই বিন্দ করার সময় আমাদের অ্যাকুরেসির কথা চিন্তা করতে হবে। আমরা চাইলেই যে ওঁ অ্যাকুরেসি পাব তার কোন গ্যারান্টি নাই।
- 50% অ্যাকুরেসির মত খারাপ আর কিছু নাই। তারমানে আমার তৈরি মডেলের পার্ফর্মেন্স স্কোর যদি ৫০% হ্যাঁ, এর মানে হল, যদি মডেলটি কোন প্রেডিকশন করে তাহলে সেটা হওয়ার সম্ভাবনা ৫০-৫০। তাই অবশ্যই আমাদের ৫০% এর বেশি অ্যাকুরেসির দিকে নজর দিতে হবে।

- আমরা যখনই রোগ প্রেডিকশন করব তখন ৫০% চরম খারাপ পার্ফর্মেন্স স্কোর। জেনেটিক ডিফারেন্স এখানে একটা বড় ফ্যাটের। দেখা গেছে জমজদের ও সব একই হল জিনগত পার্থক্য থাকেই। এই ডিম্বতার জন্য ডায়াবেটিসে আক্রান্ত হবার সম্ভাবনাও একেক জনের একেক রকম।
- সুতোঁ ৭০% অ্যাকুরেসি মোটামুটি রিজনেবল। এখন এটাকে End Point ধরে আমরা কাজ এগাতে পারি।

তাহলে চলুন আমাদের টাগেট স্টেটমেন্টে আরও কিছু পরিবর্তন আনি।

পরিবর্তিত স্টেটমেন্ট

Pima Indian Diabetes ডেটাসেট ব্যবহার করে ৭০% বা তার বেশি অ্যাকুরেসির মাধ্যমে বের করতে হবে কে কে ডায়াবেটিসে আক্রান্ত হবে

Context বা প্রাসঙ্গিক ফিল্ড

আমাদের আলোচ্য সমস্যাটি মেডিক্যাল বেজড, তাই আমাদের এখানে প্রাসঙ্গিকতা টানতে হবে। এতে করে সল্যুশন আরও ভাল হবে।

- প্রত্যেকেই জেনেটিক্যালি একে অপরের থেকে আলাদা, তাই এখানে জানা-অজানা বিভিন্ন ফ্যাটের কাজ করে। আপাতদৃষ্টিতে একই ফ্যাটের হওয়ার পরেও দুইজন লোকের একজনের ডায়াবেটিস হতে পারে এবং আরেকজনের নাও পারে।
- এই যে আমরা এখানে হতে পারা নামক সম্ভাবনাময় একটি বাক্য ব্যবহার করছি। তারমানে আমরা একদম ১০০% নিশ্চিত না, আদৌ ডায়াবেটিস হবে কি না।
- হতে পারা বা likelihood যদি আমরা আমাদের স্টেটমেন্টে অ্যাড করি তাহলে পরিবর্তিত স্টেটমেন্ট হবে,

পরিবর্তিত স্টেটমেন্ট

Pima Indian Diabetes ডেটাসেট ব্যবহার করে 70% বা তারও বেশি অ্যাকুরেসির মাধ্যমে বের করতে হবে কে কে ডায়াবেটিসে আক্রান্ত হতে পারে

সল্যুশন তৈরি করা

আমাদের স্টেটমেন্টে এখনো পর্যন্ত মেশিন লার্নিংয়ের প্রসিডিউর আসে নি। তাহলে মেশিন লার্নিং ওয়ার্কফ্লো ব্যবহার করলেই আমরা সল্যুশন তৈরির একটা ভাল ধারণা পেয়ে যাব।

- মেশিন লার্নিং ওয়ার্কফ্লো:
 - Pima Indian Data প্রিপসেসিং
 - ডেটা ট্রান্সফর্মেশন (যদি লাগে)

আবার পরিবর্তিত স্টেটমেন্ট

পরিবর্তিত স্টেটমেন্ট

মেশিন লার্নিং ওয়ার্কফ্লো ব্যবহার করে Pima Indian Data কে প্রিপ্রসেস ও প্রযোজনীয় ট্রান্সফর্মেশন করার পর একটা প্রেডিস্টিভ মডেল তৈরি করতে হবে।

এবার এই মডেলকে ৭০% বা তারও বেশি অ্যাকুরেসির সাথে নির্ণয় করতে হবে কে কে ডায়াবেটিসে আক্রান্ত হতে পারে।

ফাইনাল স্টেটমেন্ট ও প্রশ্নগুলো

- কোন ডেটাসেট ব্যবহার করতে হবে? - *Pima Indian dataset*
- টাগেট পারফর্মেন্স কত? - ৭০%
- কীভাবে সল্যুশন তৈরি করতে হবে? - *Machine Learning Workflow* ব্যবহার করে *Data preprocessing* ও *transformation* এর মাধ্যমে *predictive model*/তৈরি করতে হবে তারপর ডেটাসেট ব্যবহার করে প্রেডিস্ট করতে হবে।

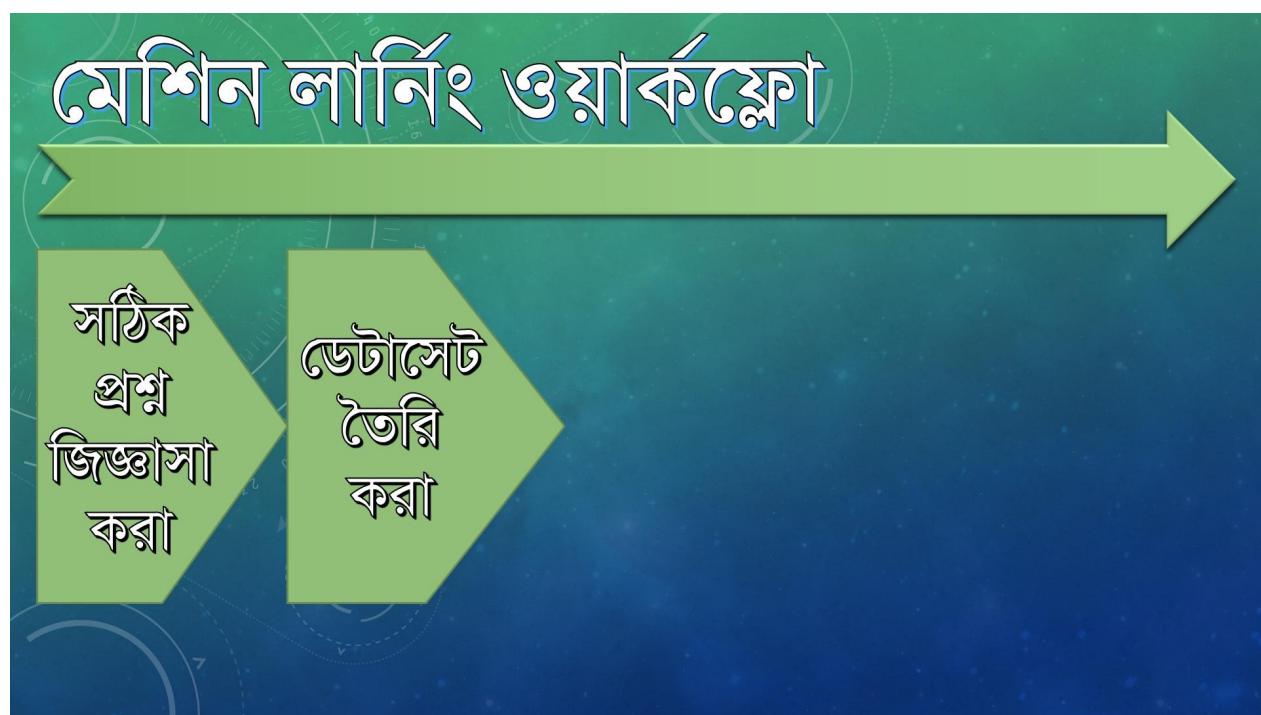
আলোচনা থেকে দেখা গেল, এক লাইনের সামান্য একটি প্রশ্ন আমরা বিভিন্নভাবে রিসার্চ করে বেশ কয়েকটি গুরুত্বপূর্ণ প্রশ্নের উত্তর পেয়ে গেছি, যেটা দিয়ে সহজেই আমরা সমস্যাটির সমাধান করতে পারি। পরবর্তী পর্বগুলোতে এই ওয়ার্কফ্লো ব্যবহার করে আমরা সমস্যাটির সমাধান করব।

"Give me six hours to chop down a tree and I will spend the first four sharpening the axe." — Abraham Lincoln

ডেটা প্রস্তুত করা (ডেটা কালেকশন ও প্রিপ্রেসিং) - ১

আমরা আগের পর্বে দেখেছি সঠিক প্রশ্নের মাধ্যমে আমাদের টাগেট স্টেটমেন্ট তৈরি করে কীভাবে।

আমরা আজকে মেশিন লার্নিংয়ের দ্বিতীয় ধাপ দেখব। দ্বিতীয় ধাপে ছিল,



মেশিন লার্নিং মানেই ডেটা নিয়ে কাজ কারবার, তাই আমি যদি বলি, ডেটা সংগ্রহ, প্রসেস করতেই মডেল বিল্ডিংয়ের সবচেয়ে বেশি সময় ব্যয় হয় সেটা আশ্চর্যের কিছু নয়।

মডেল বিল্ড হবে সংগ্রহ করা ডেটার উপর, আপনার অ্যালগরিদম যতই ভাল হোক, ডেটা যদি কার্যকর না হয় তবে আপনার প্রেডিক্টিভ মডেলও ভাল হবে না। এটা সর্বসম্মতিক্রমে স্বীকৃত। তাই মেশিন লার্নিংয়ের এই ধাপ এ আপনাকে বেশি বেশি যন্ত্রণাত হতে হবে।

আবার ডেটা প্রস্তুতকরণ যদি ভাল হয় তবে মডেল তৈরি করা অনেকটা সহজ হবে, বারবার টিউনিংয়ের দরকার হবে না এবং ডেটা ক্লিনিংয়েও প্রয়োজন হবে না। যদি ডেটা ভালভাবে প্রস্তুত না করতে পারেন, তাহলে আপনার মডেল তো ভাল হবেই না, বার বার ডেটায় হাত দিতে হবে মডেল বিল্ডিংয়ের জন্য।

তাই আগে ডেটা ক্লিন করে ম্যানেজেবল পর্যায়ে নিয়ে মডেল বিল্ডিংয়ে হাত দেওয়া ভাল।

দেখা যাক এই পর্বে আমরা কী করব।

ওভারভিউ

- ডেটা খোঁজা
- ডেটা পর্যবেক্ষণ (Inspection) ও অপযোজনীয় অংশ বাদ দেওয়া (Data Cleaning)
- ডেটা এক্সপ্লোর করা (Data Exploration)
- ডেটা মান্ডিংয়ের মাধ্যমে Tidy Data তে কনভার্ট করা
- সরকাজগুলো Jupyter Notebook এ করা

আগে দেখা যাক, Tidy Data কী?

Tidy Data

যে ডেটাসেট দিয়ে সহজে মডেল তৈরি করা যায়, সহজে ভিজুয়ালাইজ করা যায় এবং যাদের একটা নির্দিষ্ট স্ট্রাকচার বা গঠন আছে সেগুলোই হচ্ছে Tidy Data.

Tidy Data এর বৈশিষ্ট্য

- প্রত্যেকটা variable হবে একেকটা column
- প্রত্যেকটা observation হবে একেকটা row
- প্রত্যেকটা observational unit হবে একেকটা table

সংগ্রহকৃত ডেটাসেট কে Tidy ফর্মে নেয়া কিছুটা সময়সাপেক্ষ।

মেশিন লার্নিং বেজড প্রজেক্টগুলোতে ৫০-৮০% সময় ব্যয় হয় ডেটা সংগ্রহ, লিনিং আর অর্গানাইজ করতে।

ডেটা সংগ্রহ

ডেটা সংগ্রহের ভাল উৎস কোনগুলো?

- **Google**
 - গুগলে সার্চ দিলে অবশ্যই পাবেন, তবে একটু সাবধান, হাবিজাবি, ফেক আর বাতিল ডেটাও সেখানে থাকতে পারে, টেস্টিংয়ের জন্য সেগুলো ব্যবহার করা যেতেই পারে। কিন্তু কোন সিরিয়াস প্রজেক্ট করলে অবশ্যই ডেরিফাইড ডেটা সংগ্রহ করার চেষ্টা করবেন।
- সরকারি ডেটাবেজ
 - ডেটা কালকেশনের জন্য সরকারি ডেটাবেজগুলো আসলেই ভাল উৎস। কারণ এখানে আপনি মোটামুটি ডেরিফাইড ডেটাই পাবেন বলে ধরা যায়। কিছু কিছু সরকারি ডেটাবেজের সাথে ভাল ডকুমেন্টেশনও থাকে ডেটা চিনিয়ে দেওয়ার জন্য।
- প্রফেশনাল বা কোম্পানির ডেটা সোর্স
 - খুবই ভাল একটা সোর্স। বেশ কিছু প্রোফেশনাল সোসাইটি তাদের ডেটাবেজ শেয়ার করে। টুইটার তাদের

টুইট এর কালেকশন ও সেসব টুইটের নিজস্ব অ্যানালাইসিস রিপোর্ট ও শেয়ার করে থাকে। ফাইন্যাঞ্জিয়াল ডেটা পাওয়া যায় কোম্পানির ফ্রি API থেকে, যেমন Yahoo! এই ধরণের ডেটাসেট শেয়ার করে।

- আপনি যে কোম্পানিতে কাজ করবেন
 - আপনি যে কোম্পানিতে আছেন সেটাও ডেটার একটা ভাল উৎস হতে পারে।
 - ইউনিভার্সিটির ডেটা রিপোজিটরি
 - বেশ কিছু ইউনিভার্সিটি ফ্রি তে ডেটাসেট দিয়ে থাকে, যেমন University of California Irvine। এদের নিজস্ব ডেটা রিপোজিটরি আছে যেখান থেকে আপনি ডেটা কালেক্ট করতে পারবেন।
 - **Kaggle**
 - মেশিন লার্নিং নিয়ে কাজ করবেন অথচ Kaggle এর নাম জানবেন না তা হয় না। একে ডেটা সাইটস্টেডের codeforce বলতে পারেন। ডেটা অ্যানালাইসিস নিয়ে নিয়মিত কট্টেষ্ঠ হয় ওখানে। হাই গ্রেড ডেটাসেট এর জন্য অতুলনীয়।
 - **GitHub**
 - জু হাঁ, গিটহাবেও প্রচুর পরিমাণে ডেটা পাওয়া যায়। [এই Awesome Dataset Collection চেক করতে পারেন](#)
 - উপরে যেগুলো আলোচনা করা হয়েছে সবগুলাই
 - কখনো কখনো একটা সোর্সের ডেটা দিয়ে কাজ হয় না, তখন সবগুলা সোর্স ইটাই করবেন আরকি। তারপর সব ডেটা ইন্টিগ্রেট করে Tidy ডেটা বানিয়ে কাজ করতে হবে।
-

আমাদের নির্বাচিত সমস্যার ডেটাসেট কোথা থেকে সংগ্রহ করব?

Pima Indian Diabetes Data

- ডেটা ফাইল
- ডেটাসেট বিবরণ

আগেই বলা হয়েছে ডায়াবেটিস এর ডেটাবেজ আমরা সংগ্রহ করব UCI Machine Learning রিপোজিটরি থেকে।

এই ডেটাসেট এর কিছু বৈশিষ্ট্য:

- কমপক্ষে ২১ বছর বয়সের Female Patient
- ৭৬৮ টা অবজারভেশন (৭৬৮ টা Row)
 - প্রতি Row এর বিপরীতে আছে ১০ টি করে column

- ১০ টি কলামের ন টি হল Feature, মানে : Number of pregnancies, blood pressure, glucose, insuline level ... ইত্যাদি
- আর বাকি ১টা কলাম হল: ডায়াবেটিস আছে কি নাই (True / False)

এই ডেটাসেট ব্যবহার করে আমরা প্রবলেম এর সল্যুশন বের করব।

তার আগে কিছু ডেটা রূল দেখে নেয়া যাক।

Data Rule #1

আপনি যা প্রেডিষ্ট করতে চাইছেন, ডেটাসেট এ সেটা যতটা স্পষ্ট থাকবে ততটাই ভাল

কুল টা পড়তে বা শুনতে মনে হতে পারে এই কুল আর এমনকি, সাধারণ জ্ঞান দিয়েই তো বোঝা যায়।

আসলে ব্যাপারটা তা না, আমরা যেহেতু বের করতে চাচ্ছি একজন লোকের ডায়াবেটিসে আক্রান্ত হওয়ার সম্ভাবনা কত সেহেতু এই ডেটাসেট আমাদের কাজের জন্য পার্ফেক্ট, কেননা একটি কলামে ডিরেক্টলি দেওয়াই আছে, যে ব্যক্তিকে পরীক্ষা করা হয়েছে তিনি ডায়াবেটিসে আক্রান্ত কিনা?

অনেক সমস্যার সমাধান করতে গেলে আপনি ঠিক যে জিনিস টা প্রেডিষ্ট করতে চাচ্ছেন সেটা ডেটাসেট এ আলাদা করে নাও পেতে পারেন। তখন আপনাকে ডেটাসেট নতুন করে সাজাতে হবে এবং এমনভাবে সাজাতে হবে যেটা আপনার টার্গেট ড্যারিয়েবল (যে অ্যাট্ৰিবিউট প্রেডিষ্ট করবেন, যেমন এখানে ডায়াবেটিস আছে কি নাই) এর সাথে মিলে যায় বা কাছাকাছি আসে।

Data Rule #2

ডেটাসেট দেখতে যতটাই সুশ্রী মনে হোক না কেন, আপনি যেভাবে সেটা দিয়ে কাজ করতে চান সেটা কখনোই ওঁ ফরম্যাটে থাকবে না।

তাই ডেটা সংগ্রহের পরবর্তী কাজ হল ডেটা প্রিপ্রসেসিং। যেটা নিয়ে আমরা আজকে আলোচনা করব।

CSV (Comma Separated Value) ডেটা ফাইল ডাউনলোড ও নির্দেশনা

যদি আপনি UCI এর লিঙ্কে গিয়ে ডিসিট করে থাকেন তাহলে দেখবেন সেখানে .data ও .name নামের দুইটা ফাইলের লিঙ্ক দেওয়া আছে।

.data ফাইলে ড্যালুগুলো কমা সেপারেটেড আছে কিন্তু ফাইল ফরম্যাট .csv নয়, আবেকটি ব্যাপার হল সেখানে ড্যালু কোনটার মানে কী সেটাও বলা নেই (বলা আছে তবে আলাদা ফাইলে - .name)।

তাই আপনাদের কাজের সুবিধার জন্য আমি .csv ফাইলটি আপলোড করে দিয়েছি। যেখানে ড্যালুর পাশাপাশি কোন কলাম আসলে কোন প্রোপার্টি নির্দেশ করে সেটাও বলা আছে।

দুইটা ফাইল-ই ডাউনলোড করে আপনার পিসিতে রাখুন।

- csv pima dataset ডাউনলোড (original)
- csv pima dataset ডাউনলোড (modified)

নোট

- *original* : এখানে ডায়াবেচিস আছে কি নাই সেটা বলা হয়েছে 1/0 দিয়ে
- *modified* : সকল 1/0 কে TRUE/FALSE দিয়ে রিপ্রেস করা হয়েছে

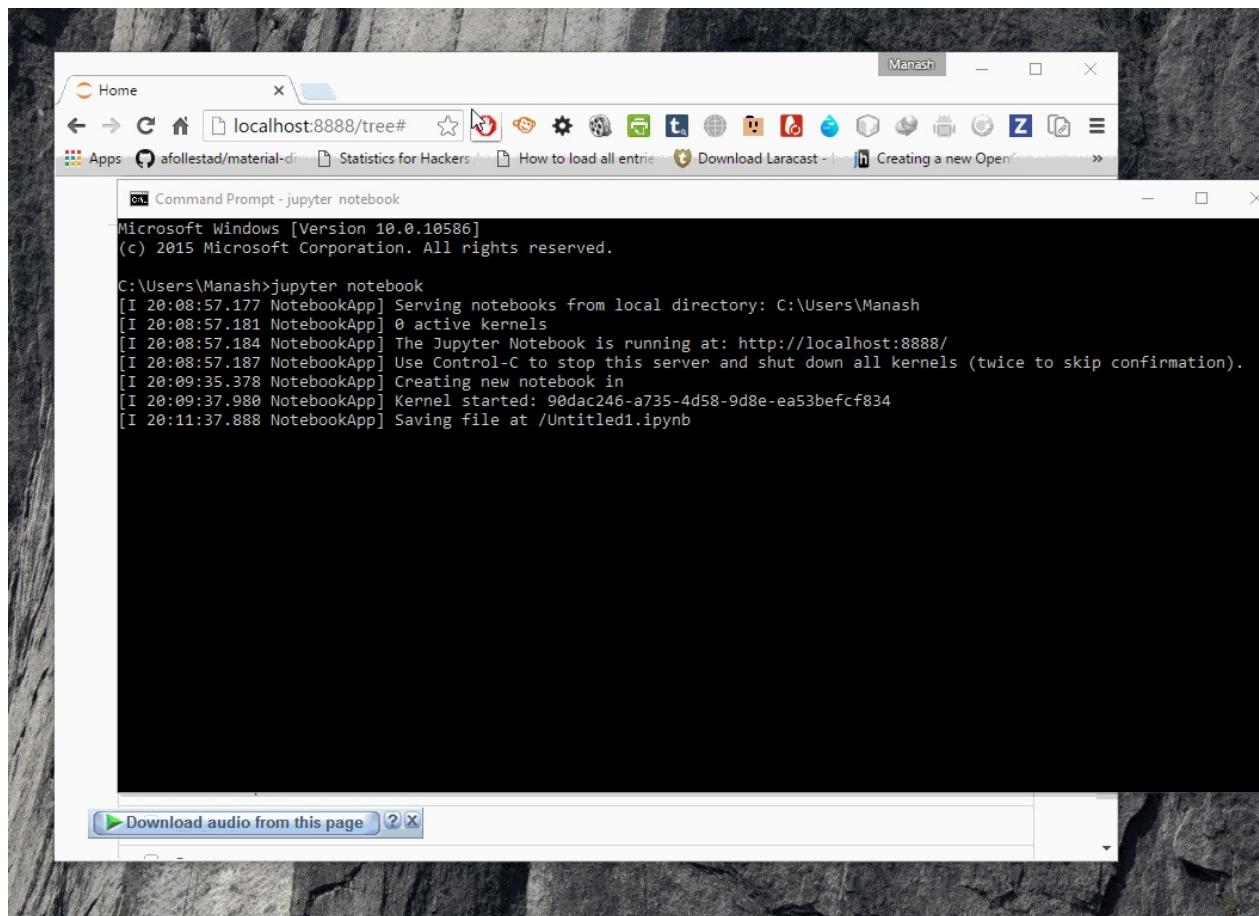
Pandas লাইব্রেরি দিয়ে ডেটা এক্সপ্লোরেশন

ipython notebook সম্পর্কে কিছুটা জেনেছেন তো? না জেনে থাকলে [এখান থেকে](#) একবার দেখে নিন।

- উইন্ডোজে থাকলে cmd ওপেন করুন ও নিচের কমান্ড দিয়ে নোটবুক ওপেন করুন
 - ipython notebook
 - যদি ওঁ কমান্ড না কাজ করে তাহলে এটা ট্রাই করুন jupyter notebook



- আপনার ব্রাউজার ওপেন হলে New > Python 2 একটি পাইথন ফাইল ওপেন করুন আর এখানে দেখানো কাজগুলো করে ফেলুন।



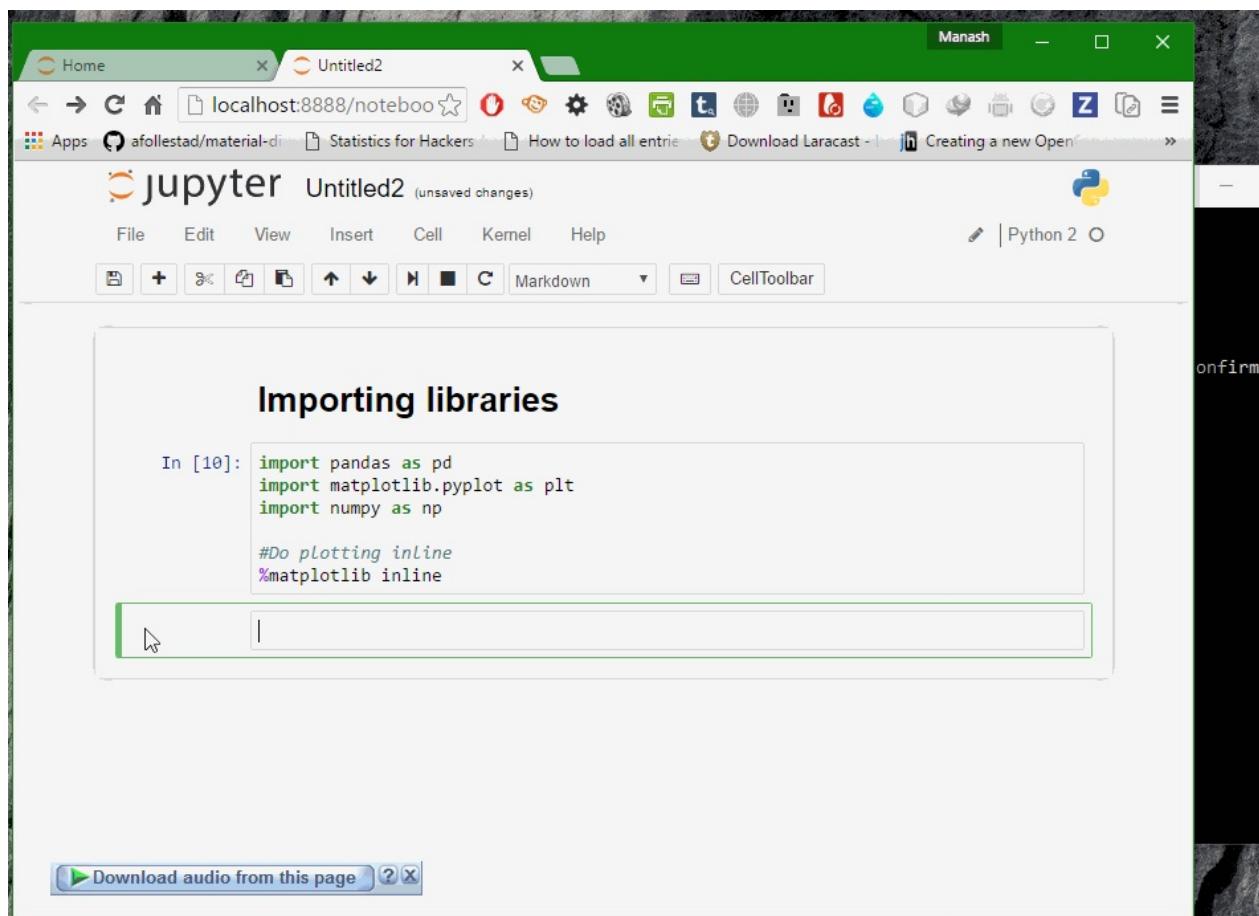
প্রয়োজনীয় লাইব্রেরি ইম্পোর্ট

কাজ শুরু করার আগে নিচের কোড দিয়ে আমরা প্রয়োজনীয় লাইব্রেরি গুলো অ্যাড করে নিলাম

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

#ইনলাইন স্লটিংয়ের জন্য জুটিওর নোটবুকের ম্যাত্রিক ফাঁগন (আলাদা উইল্টচারে স্লট রেগ করতে চাচ্ছ না আমরা)
%matplotlib inline
```

ডেটা লোড ও রিভিউ



`pd.read_csv('file_path')`

আমরা এখানে `pandas` লাইব্রেরি কে `pd` হিসেবে (`as`) ইম্পোর্ট করেছি, তারমানে `pandas` এর কোন ফাংশন কল করার জন্য আমার `pandas` কথাটা পুরা লেখার দরকার নাই, `pd` লিখলেই হবে।

আমি যদি এটা করতাম,

```
import pandas as PANDA
```

তাহলে ফাংশন কল করার জন্য `PANDA.read_csv('file_path')` এভাবে লিখতাম।

এবাব আসি `read_csv` ফাংশনে, ফাংশন থেকে বোঝা যায় এর কাজ হচ্ছে `csv` ফাইল রিড করা।

এই ফাংশন `csv` ফাইলকে কনভার্ট করে `Pandas` ডেটাফ্রেম ফরম্যাটে পরিণত করে। যেটার বিভিন্ন ধরণের পরিবর্তন আমরা `Pandas` লাইব্রেরি দিয়েই করতে পারব।

`read_csv('filePath')` এখানে আমি আগুমেন্টে আমার পিসির যেখানে `csv` ফাইল ছিল সেই ডিরেক্টরি দিয়েছি। আপনার ক্ষেত্রে অবশ্যই আপনার পিসির যেখানে ফাইলটা আছে সেই ডিরেক্টরি দিতে হবে।

`data_frame.shape`

ডেটাফ্রেমের ডেটা যেহেতু একটি ম্যাট্রিক্স (বা 2D Array) তাই আমরা এর Row আর Column সংখ্যা দেখাব জন্য shape ভ্যারিয়েবলটি কল করেছি।

আউটপুট Row - 768 (লেবেল ছাড়া) আর Column - 10 টা

data_frame.head(number)

data_frame.head(3) ফাংশনটি কল করার মাধ্যমে আমরা ডেটাফ্রেমের প্রথম 3 টি রো প্রিন্ট করলাম।

data_frame.tail(number)

data_frame.tail(4) ফাংশনটি কল করার মাধ্যমে আমরা ডেটাফ্রেমের শেষের 4টি Row প্রিন্ট করলাম।

আজকের চ্যাপ্টার এখানেই শেষ, তবে এটা ডেটা প্রিপ্রসেসিংয়ের প্রথম অংশ ছিল। পরবর্তী পর্বে আমরা ডেটা প্রিপ্রসেসিংয়ের ফান্ডামেন্টাল বিষয় গুলো নিয়ে আলোচনা করব।

"Organize, don't agonize." — Nancy Pelosi

ডেটা প্রস্তুত করা (ডেটা প্রিপ্রেসিং) - ২

ডেটাফ্রেম পরিবর্তন করা

প্রায় সময়ই ডেটাসেটে ডেটা মিসিং থাকতে পারে। আমাদের সেই মিসিং ডেটাও হ্যান্ডেল করতে হবে। যাঁ, হ্যাত আমরা হারানো ডেটা পাব না, তবে প্রয়োজনীয় ব্যবস্থা না নিলে প্রোগ্রাম ক্র্যাশ করতে পারে।

কোন কোন Column বাদ দিতে হবে?

- যেগুলো ব্যবহার করা হবে না
- কলাম আছে কিন্তু ডেটা নাই
- একই কলাম যদি একাধিকবার থাকে, তাহলে একটা রেখে বাকিগুলো মুছে দিতে হবে
 - অনেক সময় নাম দেখে মনে হতে পারে দুইটা আলাদা কলাম কিন্তু আসলে জিনিসটা একই। উদাহরণ হিসেবে বলা যায়, একটা কলামে লেখা আছে Length (meter) এবং আরেকটি কলামে লেখা আছে Size (centimeter), হঠাৎ দেখলে মনে হবে দুইটা জিনিস আলাদা কারণ লেবেল হচ্ছে Size ও Length। কিন্তু ভাল করে লক্ষ করে দেখা গেল, Length এর প্রত্যেকটা ডেটাকে 100 দিয়ে গুণ করে আমরা Size এর ডেটাগুলি পেয়ে যাচ্ছি। হাতে ক্যালকুলেশন করে একই ধরণের ডেটা বের করা সম্ভব হ্যাঁ না এবং হলেও এটা কোন এফিশিয়েন্ট পদ্ধতি না। এই অতিরিক্ত কলামগুলো আসলে ডেটাসেট এন্যেজ জেনারেট করে। আমরা স্ট্যাটিস্টিক্যাল অ্যানালাইসিস (এখানে Correlation) এর মাধ্যমে একই রকম কলামগুলি আলাদা করব।

Correlated Column কী?

- একই তথ্য যদি একটু ভিন্ন ফরম্যাটে থাকে, উপরের উদাহরণে Length এবং Size আসলে একই জিনিস, শুধু Unit আলাদা। তারমানে এরা Correlated Column।
- অন্ন ইনফরমেশন অ্যাড করে বা করেই না।
- লার্নিং অ্যালগরিদমকে কনফিউজ করে।

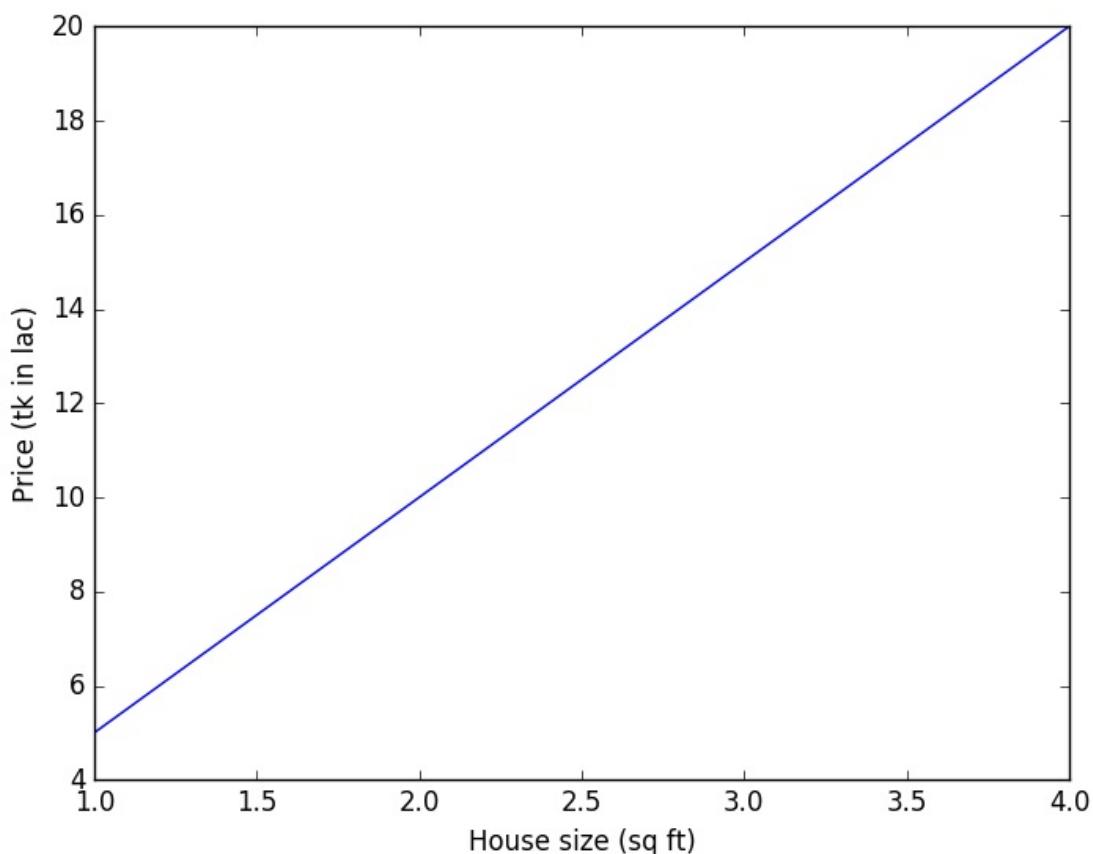
লিনিয়ার রিগ্রেশন নিয়ে অন্ন কিছু কথা

পরবর্তী উদাহরণ বুঝতে গেলে আমাদের লিনিয়ার রিগ্রেশন এর কিছু বেসিক লাগবে।

নিচের কান্সিক ডেটাসেট এর কথা চিত্তা করা যাক,

House Size (sq ft)	Price (Tk in lac)
1	5
2	10
3	15
4	20

গ্রাফ



আপনাকে যদি বলা হয়, 5 sq ft বাড়ির দাম কত হবে? আপনি নির্দিষ্য বলে দিতে পারবেন, উত্তর হবে 25 lac ।
কীভাবে বললেন?

খুব সহজ, প্রতি 1 sq ft বৃদ্ধির জন্য দাম বাঢ়ছে 5 lac করে।

আমরা যদি একটা ম্যাথেমেটিক্যাল মডেল দাঁড়া করাতে চাই, সেটা হবে অনেকটা এরকম।

$$price = size(sqft) \times 5(tkinlac)$$

বা,

$$y = f(x) = \alpha \times x$$

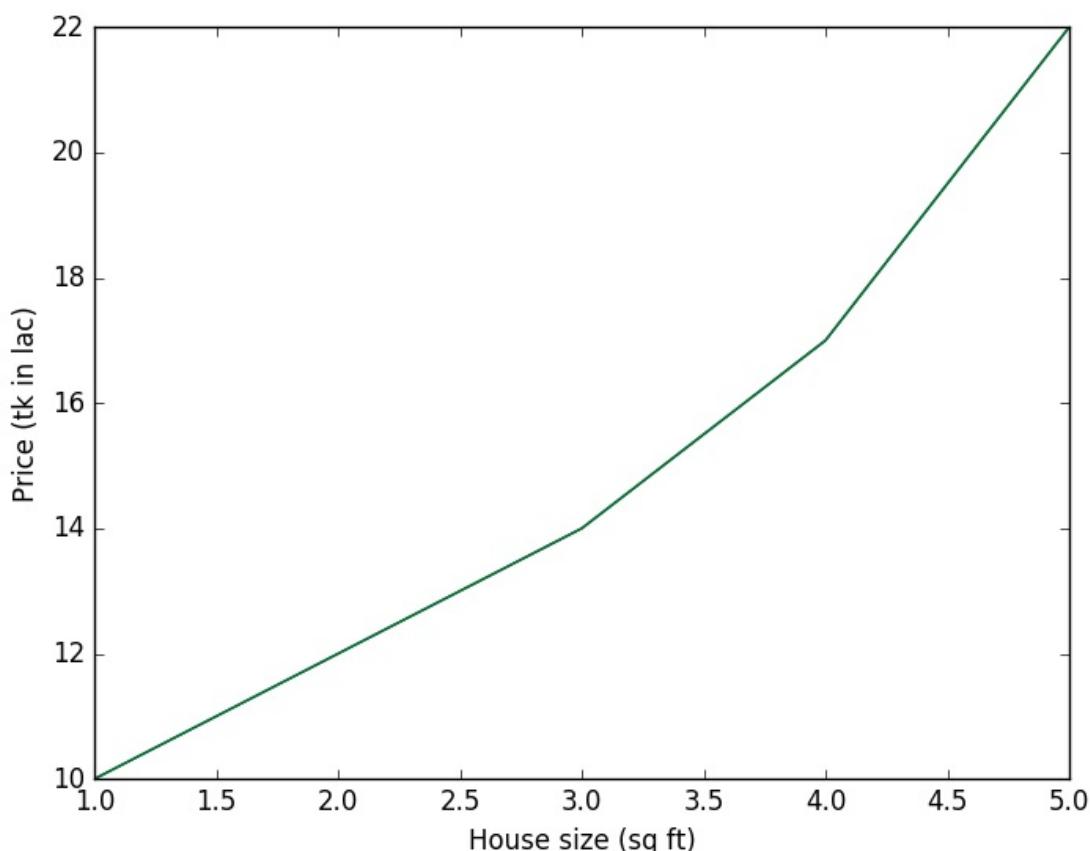
যেখানে, y হচ্ছে প্রাইস, x হচ্ছে সাইজ α হচ্ছে 5 এবং $f(x)$ ফাংশনটি বলে দিচ্ছে x এর মানের জন্য প্রাইস কত হবে

বাস্তবে মডেল এতটা সহজ হয় না, অনেক প্যাঁচ থাকে, এখন আমি একটা α গুণ দিয়েই মান পেয়ে যাচ্ছি তখন β , γ , θ হাবিজাবি যা আছে তা দিয়ে গুণ দিলেও হয়ত কাছাকাছি মান পাবেন না।

নিচের ডেটাসেট দেখা যাক,

House Size (sq ft)	No of rooms	Price (tk in lac)
1	3	10
2	3	12
3	4	14
4	4	17
5	5	22

গ্রাফ



এবার আপনাকে যদি বলি, বাড়ির আকার যদি 6 sq ft হয় তাহলে প্রাইস কত হবে? এবার আপনি বেশ ঝামেলায় পড়ে যাবেন, কারণ প্রতি ক্ষয়ার ফিট আকার বৃদ্ধির সাথে বর্ধিত দাম সুব্যবস্থা নয়। আগেরটা বিয়োগ দিয়ে পার্থক্য বের করে সেটার সাথে পার্থক্য যোগ করে পরের প্রাইস পেয়ে যাবেন, সমস্যাটা এত সহজ নয়। কারণ সাথে আবার যুক্ত হয়েছে

No of rooms |

এখন যদি আমাকে বলা হয়, এটার একটা ম্যাথেমেটিক্যাল মডেল দাঁড় করতে তাহলে আমিও বেশ ঝামেলায় পড়ে যাব। এমন কোন সেই লিনিয়ার ইকৃয়েশন, যেটাতে 1, 2, ... 5 ইনপুট দিলে যথাক্রমে 10, 12 ... 22 পাওয়া যায়?

এক্স্যাট কোন মডেল বিন্দ না করতে পারলেও হ্যত কাছাকাছি কোন মডেল তৈরি করতে পারব যার ইকৃয়েশন অনেকটা এরকম হতে পারে,

$$price = \alpha \times size + \beta \times rooms$$

Correlated Column এর উদাহরণ

ধৰা যাক, আমরা আবারও সেই বিখ্যাত সমস্যা House Price Prediction টা আলোচনায় আনি।

House Area (Acre)	Size (kilo sq meter) (approx.)	No of rooms	Price (tk in lac)
1	4	3	10
2	8	4	12
3	12	4	16

ডেটাসেট এর কলাম ভালভাবে পরীক্ষা না করেই প্রেডিষ্ট করতে বসে গেলাম নিচের ফরমুলা (Linear Regression ফরমুলা) দিয়ে,

$$Price = \alpha * Area (Acre) + \beta \times Size(kilo sq meter) + \gamma \times noOfRooms$$

আমরা লিনিয়ার রিগ্রেশনের ক্ষেত্রে দেখেছিলাম প্রত্যেকটা ফিচার (ইনপুট ড্যারিয়েবল) কে একটা Co-efficient দিয়ে গুণ করি তারপর সেগুলোকে যোগ করে আউটপুট প্রেডিষ্ট করি। একই রকম কলাম Area & Size দুইবার রাখার কারণে আউটপুট Price কখনোই ঠিকঠাক আসবে না।

এখনে কলাম দুইটা একই রকম সেটা সহজে বোঝা যাচ্ছে কারণ উদাহরণটা আমার তৈরি করা :P। জোকু অ্যাপার্ট, যদি অনেকগুলো কলাম হয়, আর সবগুলোর নাম আলাদা হয় আর ডেটাও আলাদা হয় কিন্তু আসলে একটা আরেকটার ইউনিট বেজড সিনোনিম হয় সেগুলো বের করা অনেক জটিল ব্যবহার। তাই আমরা এখনে পরিসংখ্যানের একটি গুরুত্বপূর্ণ টপিক (Correlation) এর সাহায্য নেব।

Pearson's Correlation Co-efficient বা Pearson's r

Pandas লাইব্রেরিতে কো-রিলেশন ফাংশন কল করলে সেটা নিচের সূত্রানুযায়ী কো-রিলেশন ক্যালকুলেট করে। কো-রিলেশন নিয়ে বিশ্লেষণ আলোচনা আরও করা হবে, আপাতত এই ফরমুলা নিয়ে খুশি থাকুন।

$$r = \frac{\sum xy}{\sqrt{\sum x^2 \sum y^2}}$$

এই ফরমুলায়, x হচ্ছে একটা ভ্যারিয়েবল আৰু y হচ্ছে আৰেকটা ভ্যারিয়েবল (Isn't it too obvious?)।

আমাদেৱ বেৱ কৰতে হবে r এৰ মান কত। r এৰ মান দিয়ে আমৰা বুঝতে পাৰি যে দুইটা ভ্যারিয়েবলেৰ সামঞ্জস্যতা কতখনি। যদি $r = 1$ হয় তাৰমানে দুইটা ভ্যারিয়েবলেৰ মধ্যে কোন পাৰ্থক্য নাই, তাই যেকোন ভ্যারিয়েবলেৰ নিজেৰ সাথে কো-ৱিলেশন ক্যালকুলেট কৰলে r এৰ মান হয় 1।

আৱও ব্যাখ্যা যদি চান, উপৰেৰ উদাহৰণেৰ Acre এবং Sq meter এৰ মধ্যকাৰ Correlation Co-efficient ক্যালকুলেৰ কৰলে r এৰ মান 1 পাৰেন।

প্ৰমাণ:

Pearson Correlation Coefficient Calculator

ERROR! (See below.)

X Values	Y Values
1 2 3	4 8 12

Error! The sample size is too small to allow a reliable calculation of the Pearson correlation coefficient. Please input additional data.

এবাৰ দেখা যাক ডেটাসেটেৰ কোন কলামে কোন ডেটা মিসিং আছে কিনা সেটা কীভাৱে বেৱ কৰা যায়।

Null বা ডেটাসেট এৰ ফাঁকা অংশ বেৱ কৰা

আগেৱ তৈৰি কৰা নোটবুক ওপেন কৰন আৰু নিচেৰ কোডটি লিখুন,

```
print data_frame.isnull().values.any()
```

isnull().values.any()

`isnull()`

এটা আবার সেই ডেটাফ্রেমকেই রিটার্ন করে কিন্তু পার্থক্য হল সেখানে আর ভ্যালু থাকে না, Empty Cell রিপ্রেস হয় `True` দিয়ে আর Non-Empty Cell রিপ্রেস হয় `False` দিয়ে।

In [12]: `data_frame.head(3)`

Out[12]:

	num_preg	glucose_conc	diastolic_bp	thickness	insulin	bmi	diab_pred	age	s
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	1
2	8	183	64	0	0	23.3	0.672	32	0

In [13]: `data_frame.tail(4)`

Out[13]:

	num_preg	glucose_conc	diastolic_bp	thickness	insulin	bmi	diab_pred	age
764	2	122	70	27	0	36.8	0.340	27
765	5	121	72	23	112	26.2	0.245	30
766	1	126	60	0	0	30.1	0.349	47
767	1	93	70	31	0	30.4	0.315	23

Checking for null data

In []:

`values.any()`

755	False	F							
756	False	F							
757	False	F							
758	False	F							
759	False	F							
760	False	F							
761	False	F							
762	False	F							
763	False	F							
764	False	F							
765	False	F							
766	False	F							
767	False	F							

768 rows × 10 columns

In []:

[

`isnull()` রিটার্ন করে ডেটাফ্রেম, কিন্তু `.values` দিলে সেটা `True/False` এর একটা অ্যারে তে পরিণত হয়।

`.any()` ফাংশন চেক করে অ্যারেতে থাকা কোন ড্যালু ফাঁকা বা Empty কিনা।

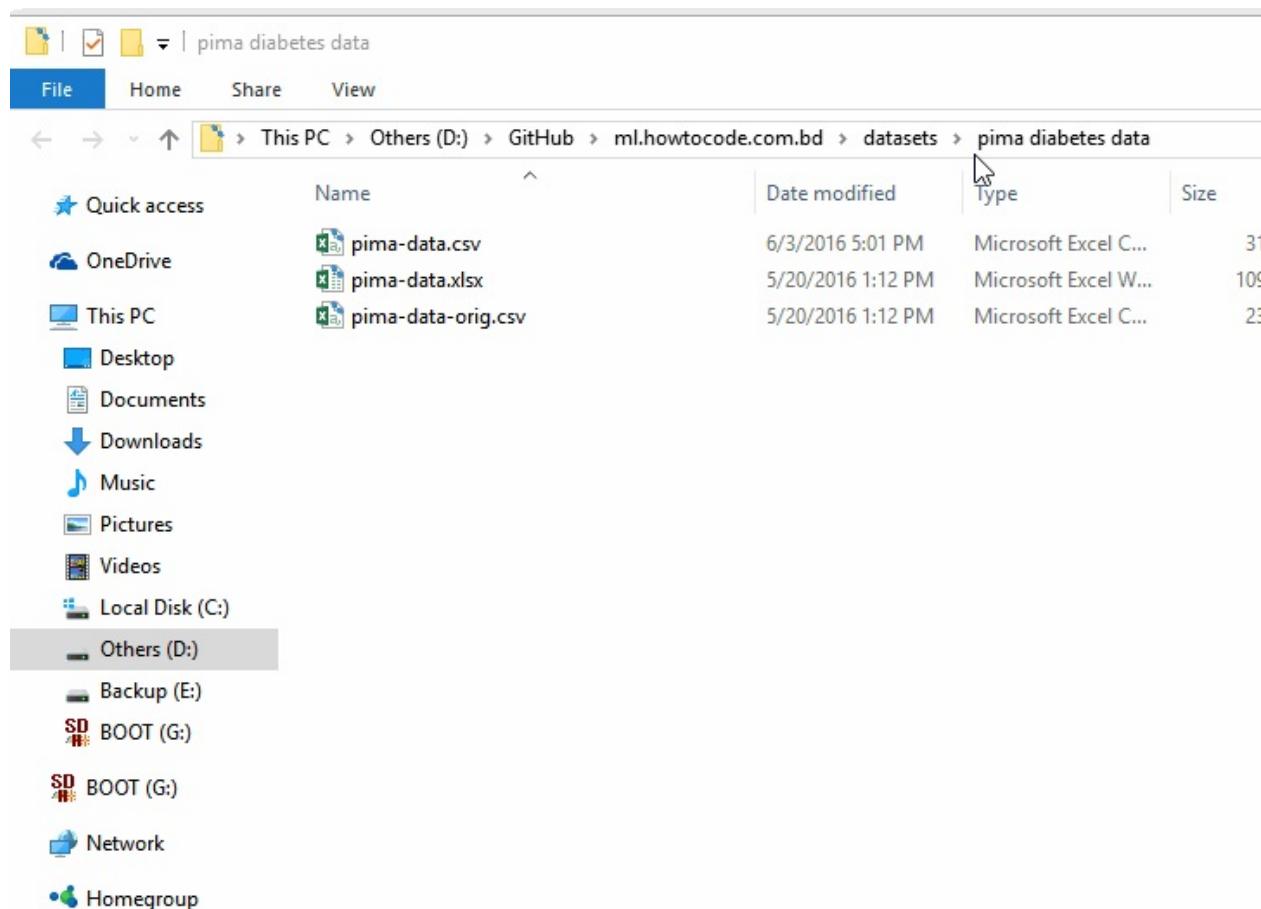
`pima-data.csv` ফাইলে কোন ফাঁকা ডেটা নাই। তাই এই প্রোগ্রাম স্টেটমেন্টটি কল করলে `False` দেখায়।

ইচ্ছাকৃত একটা **Cell** ডিলেট করে আবার

`data_frame.isnull().values.any()` স্টেটমেন্ট রান করা

এখনে আমি `pima-data.csv` ফাইলের একটা সেল ইচ্ছে করে ডিলেট করে আবার Pandas দিয়ে লোড করে কোডটা চালিয়ে দেখলাম।

দেখা যাচ্ছে এখন আউটপুট আসছে `True`। তারমানে কোন না কোন একটা সেল খালি আছে।



Correlation Matrix Heatmap তৈরি করা

আমরা এতক্ষণে Correlation সম্পর্কে কিছুটা জানলাম আর দেখলাম ডেটাসেট এ কোন Null ভ্যালু লুকিয়ে থাকলে সেটাকে কীভাবে বের করা যায়। এখন দেখব, কীভাবে Correlation Matrix Heatmap জেনারেট করতে হয়। তার আগে একটু বলা যাক, Heatmap টা কী জিনিস।

Heatmap

উইকিপিডিয়া অনুসারে,

A heat map (or heatmap) is a graphical representation of data where the individual values contained in a matrix are represented as colors.

অর্থাৎ, নিউমেরিক্যাল ভ্যালু আমরা রং দিয়ে রিপ্রেজ করে একটা প্লট জেনারেট করি। সেটাই হবে Heatmap।

তারমানে, Correlation Heatmap হচ্ছ Correlation ভ্যালুগুলোকে রং দিয়ে রিপ্রেজ করে গ্রাফে প্লট করা।

Correlation Heatmap

আমরা দেখেছি, দুইটা ভ্যারিয়েবলের মধ্যে [কো-রিলেশন ক্যালকুলেট](#) করে কীভাবে

আপনি নিজেই নিজেকে প্রশ্ন করে দেখুন, কতগুলি ড্যালু (ফ্লোটিং পয়েন্ট) কে তুলনা করা সহজ নাকি রং তুলনা করা সহজ? অবশ্যই রং তুলনা করা সহজ,

আমাদের যে কাজটা করতে হবে সেটা হল একটা ড্যারিয়েবল বাছাই করে প্রত্যেকটা ড্যারিয়েবলের সাথে কো রিলেশন বের করতে হবে (এমনকি তার নিজের সাথেও)। এটা করার জন্য আমরা ড্যারিয়েবল গুলো Row এবং Column wise সাজাব,

----	num_preg	glucose_conc	diastolic_bp	thickness	insu
num_preg	1	corr_value	corr_value	corr_value	corr_v
glucose_conc	corr_value	1	corr_value	corr_value	corr_v
diastolic_bp	corr_value	corr_value	1	corr_value	corr_v
thickness	corr_value	corr_value	corr_value	1	corr_v
insulin	corr_value	corr_value	corr_value	corr_value	1
bmi	corr_value	corr_value	corr_value	corr_value	corr_v
age	corr_value	corr_value	corr_value	corr_value	corr_v

আগেই বলা হয়েছিল, কোন ড্যারিয়েবলের নিজের সাথে কো রিলেশন সবসময় 1 হবে। টেবিলের ডায়াগনাল বরাবর যত মান আছে সব অবশ্যই 1 হবে কারণ তাদের নিজেদের মধ্যে কো-রিলেশন বের করা হয়েছে। আর corr_value দ্বারা বুঝানো হয়েছে একটা ড্যারিয়েবল ও আরেকটা ড্যারিয়েবলের কো-রিলেশন কোন একটা ড্যালু হতে পারে, যেহেতু আমরা লাইব্রেরি ব্যবহার করে এই ড্যালুগুলং নির্ধারণ করব তাই আমাদের নিজেদের হাতে ক্যালকুলেট করার প্রয়োজন দেখছি না।

এবার যেটা গুরুত্বপূর্ণ কাজ সেটা হল হিটম্যাপের রং বাছাই করা। চিন্তা করার কিছু নাই, Matplotlib লাইব্রেরির বিল্ট ইন কালার ম্যাপ দেখেই আমরা আপাতত কাজ করতে পারব। আপনি চাইলে ডকুমেন্টেশন ঘেঁটে নিজের পছন্দমত রং দিতে পারেন। আপাতত আমরা ডিফল্টটাই ব্যবহার করব।

Matplotlib Heat Map Color Guide

Matplotlib হিটম্যাপ জেনারেট করার সময় নিচের সিকোয়েন্স অনুযায়ী রং সেট করবে।

Less Correlated to More Correlated
Blue -> Cyan -> Yellow -> Red -> Dark Red (Correlation 1)

Heatmap জেনারেট করার ফাংশন

চলুন, চটপট হিটম্যাপ জেনারেট করার ফাংশন লিখে ফেলি, ফাংশনটা হবে এরকম

```

# Here size means plot-size
def corr_heatmap(data_frame, size=11):
    # Getting correlation using Pandas
    correlation = data_frame.corr()

    # Dividing the plot into subplots for increasing size of plots
    fig, heatmap = plt.subplots(figsize=(size, size))

    # Plotting the correlation heatmap
    heatmap.matshow(correlation)

    # Adding xticks and yticks
    plt.xticks(range(len(correlation.columns)), correlation.columns)
    plt.yticks(range(len(correlation.columns)), correlation.columns)

    # Displaying the graph
    plt.show()

```

কেন সাবপ্লট ব্যবহার করলাম?

ইচ্ছা করলে এখানে `plt.matshow(correlation)` ব্যবহার করেও হিটম্যাপ জেনারেট করা যেত, কিন্তু তাতে আমি ইচ্ছামত আকারের গ্রাফ জেনারেট করতে পারতাম না, তাই প্লটকে সাবপ্লটে নিয়ে সাইজ অ্যাসাইন করে ইচ্ছামত আকারের সুবিধাজনক প্লট জেনারেট করা যাচ্ছে।

`xticks` ও `yticks` কী?

`plt.xticks(range(len(correlation.columns)), correlation.columns)` এই কোড দিয়ে বুঝানো হয়েছে, প্রতি ব্লকের দৈর্ঘ্য হবে 1 একক করে এবং দাগগুলো হবে 0, 1, 2 ... `len(correlation.columns)` পর্যন্ত। আর পরবর্তী আণুমন্তি `(correlation.columns)` দিয়ে প্রতিটা ব্লকের লেবেল দেওয়া হয়েছে।

`plt.yticks..` এর জন্য একই কথা প্রযোজ্য।

`plt.show()` দিয়ে কী করা হয়েছে?

U kiddin' bro?

`corr_heatmap(data_frame, size)` ফাংশনের মাধ্যমে হিটম্যাপ প্লটিং

কষ্ট করে ফাংশন লিখলাম আর না ব্যবহার করলে চলে? নিচের কোড নিপেট দিয়ে সহজেই হিটম্যাপ প্লট করতে পারি,

```
corr_heatmap(data_frame)
```

```
In [17]: data_frame.isnull().values.any()
Out[17]: False

In [22]: data_frame = pd.read_csv('D:\GitHub\ml.howtocode.com.bd\datasets\pima diabetes data\pima
In [23]: data_frame.isnull().values.any()
Out[23]: True

In [39]: def corr_heatmap(data_frame, size=11):
    # Getting correlation using Pandas
    correlation = data_frame.corr()

    # Dividing the plot into subplots for increasing size of plots
    fig, heatmap = plt.subplots(figsize=(size, size))

    # Plotting the correlation heatmap
    heatmap.matshow(correlation)

    # Adding xticks and yticks
    plt.xticks(range(len(correlation.columns)), correlation.columns)
    plt.yticks(range(len(correlation.columns)), correlation.columns)

    # Displaying the graph
    plt.show()
```

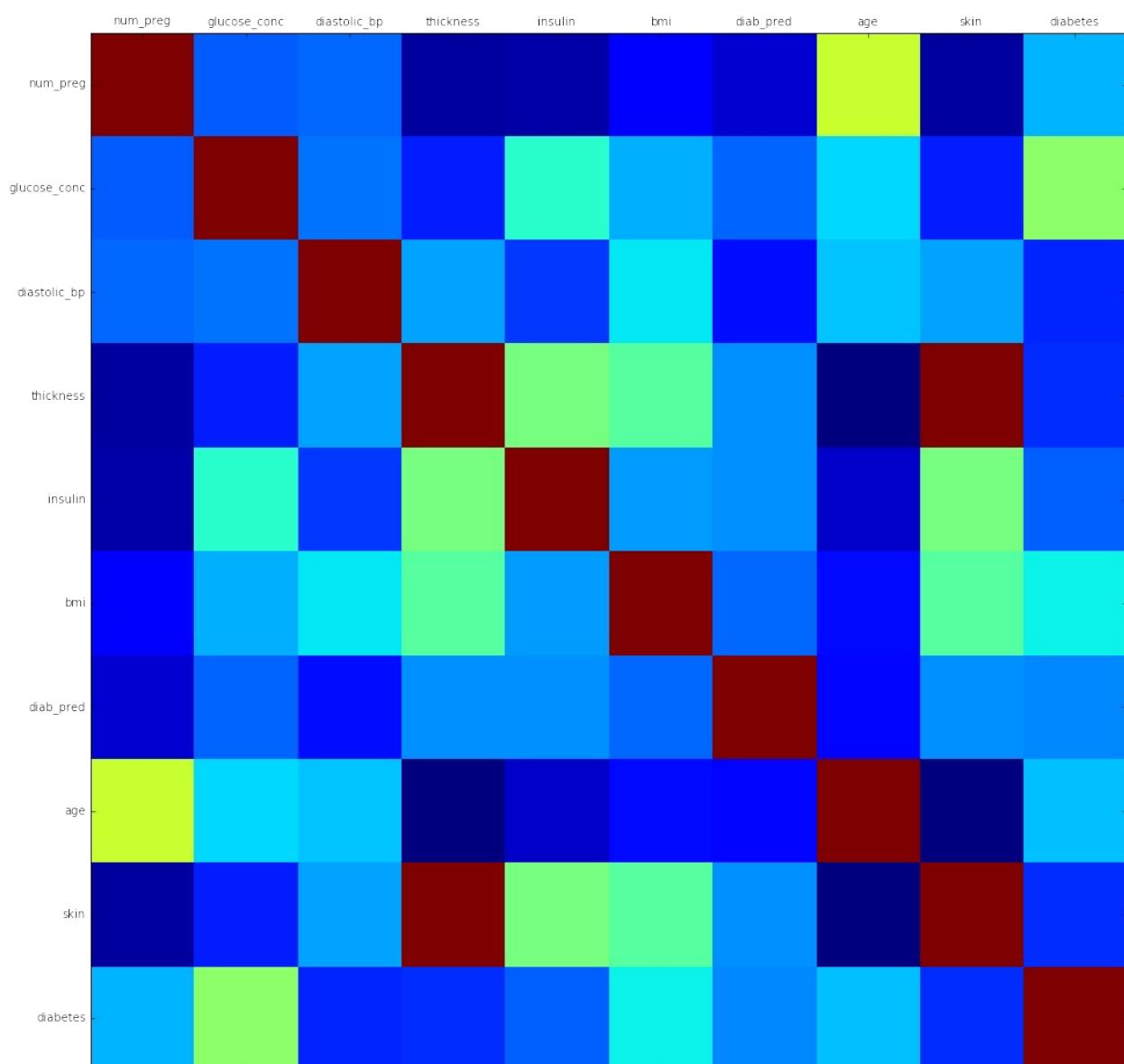
In []:

In []:

In []:



জেনারেটেড হিটম্যাপ ক্লোজডিউ



লক্ষণীয়

আমরা আগেই দেখেছিলাম দুইটা ড্যারিয়েবল যদি একই রকম হয় তাহলে তাদের Correlation 1 হবে। ডায়াগনালে প্রতিটা ড্যারিয়েবলে তার নিজের সাথে কো-রিলেশন বের করা হয়েছে তাই ডায়াগনালের স্কেলগুলোর রং গাঢ় লাল।

কিন্তু ভাল করে লক্ষ করে দেখবেন, `skin` এবং `thickness` এই দুইটার কো-রিলেশন কিন্তু 1 (গাঢ় লাল রং)।

তারমানে, `skin` আর `thickness` আসলে একই জিনিস, একক এবং হ্রফের হয়েছে শুধু। বিশ্বাস হচ্ছে না?

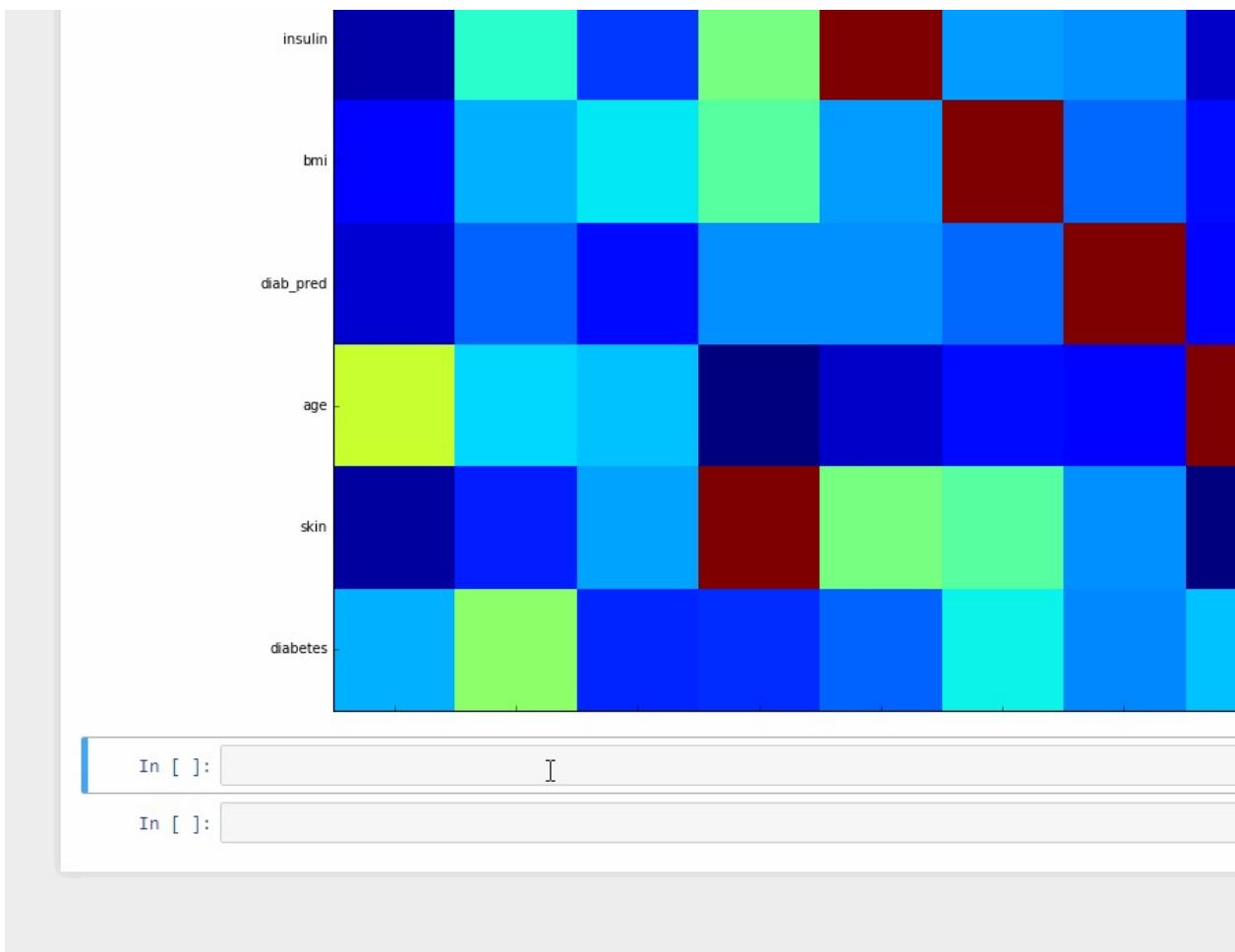
এক কাজ করুন তাহলে, `thickness` এর প্রতিটা ড্যালু কে `0.0393701` দিয়ে গুণ দিন তাহলে দেখবেন আপনি `skin` এর ড্যালু পেয়ে যাচ্ছেন। `1 millimeter = 0.0373701 inch` এবার আপনিই বলতে পারবেন কোনটার একক আসলে কী?

কালপ্রিট পেলাম, এবার ডেটাসেট ক্লিনিং

উপরের কাজ থেকে এটা বুঝলাম আমরা যে একই টাইপের কলাম কোনগুলা। Tidy Data এর বৈশিষ্ট ছিল প্রতিটা কলাম কে অবশ্যই Unique হতে হবে। ডুপ্লিকেটগুলো থেকে একটা রেখে বাকিটা ডেটাসেট থেকে উৎখাও করতে হবে।

আমি এখানে skin ড্যারিয়েবল উৎখাও করব, আপনি চাইলে একে অথবা thickness কে উৎখাও করতে পারেন, সম্পূর্ণ আপনার ইচ্ছা।

```
# Deleting 'skin' column completely
del data_frame['skin']
# Checking if the action was successful or not
df.head()
```



In []:

In []:

আমরা একটা ডুপ্লিকেট কলাম কে ফেলে দিতে পারলাম। এখনো কাজ শেষ হয় নাই, ডেটা মোন্ট করতে হবে। চিন্তার কিছু নাই, ডেটা প্রিপারেশনের এটাই শেষ ধাপ। So cheers!

ডেটা মোন্টিং (Data Molding)

ডেটা টাইপ অ্যাডজাস্টমেন্ট

আমাদের ডেটাসেট এমন হতে হবে তা যেন সবরকম অ্যালগরিদমে কাজ করার উপযোগী হয়। না হলে প্রতিটি অ্যালগরিদমের জন্য আমাদের ডেটা টুইকিং করতে হবে যেটা বেশ ঝামেলার কাজ। তাই আমরা ঝামেলার কাজটা বার বার না করে একবারই করব যাতে আর সেটা মাথাব্যাথার কারণ না হয়ে দাঁড়ায়।

ডেটা টাইপ চেকিং

ডেটা মোড়িংয়ের আগে একবার ডেটাটাইপ গুলো চেক করে নেওয়া যাক।

```
data_frame.head()
```

এটা দিলেই আবারও ডেটাফ্রেমের কিছু স্যাম্পল দেখতে পাবেন এবং ভাল করে লক্ষ করে দেখবেন এখানে সবগুলো ড্যালুই ফ্লোট বা ইন্টিজার টাইপ কিন্তু একটা রয়ে গেছে Boolean টাইপ।

	num_preg	glucose_conc	diastolic_bp	thickness	insulin	bmi	diab_pred	age	diabetes
0	6	148	72	35	0	33.6	0.627	50	True
1	1	85	66	29	0	26.6	0.351	31	False
2	8	183	64	0	0	23.3	0.672	32	True
3	1	89	66	23	94	28.1	0.167	21	False
4	0	137	40	35	168	43.1	2.288	33	True

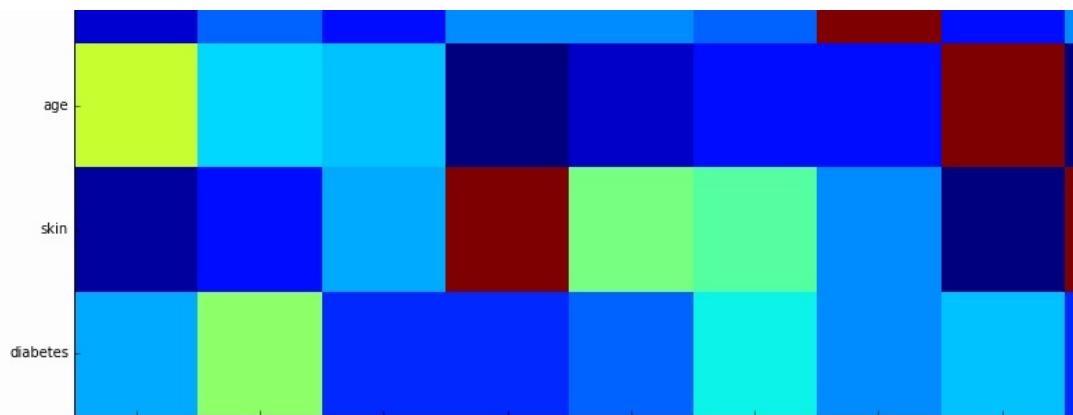
ডেটা টাইপ চেঙ্গিং

True কে আমরা 1 বানাবো এবং False কে বানাব 0। নিচের কোড স্লিপেট টি দিয়েই কাজটা করা যাবে,

```
# Mapping the values
map_diabetes = {True : 1, False : 0}

# Setting the map to the data_frame
data_frame['diabetes'] = data_frame['diabetes'].map(map_diabetes)

# Let's see what we have done
data_frame.head()
```



```
In [58]: del data_frame['skin']
```

```
In [59]: data_frame.head()
```

Out[59]:

	num_preg	glucose_conc	diastolic_bp	thickness	insulin	bmi	diab_pred	age	diabetes
0	6	148	72	35	0	33.6	0.627	50	True
1	1	85	66	29	0	26.6	0.351	31	False
2	8	183	64	0	0	23.3	0.672	32	True
3	1	89	66	23	94	28.1	0.167	21	False
4	0	137	40	35	168	43.1	2.288	33	True

```
In [ ]: map_diabetes = {True :
```

অভিনন্দন!

এই মোডেড ও লিনড ডেটাসেট আমরা আমাদের ইচ্ছানুযায়ী অ্যালগরিদমে বসিয়ে কাজ করতে পারবো।

কিন্তু?

Data Rule #3

Rare ইডেন্ট হাই অ্যাকুরেসির সাথে প্রেডিষ্ট করার সম্ভাবনা কম

স্বাভাবিক, কারণ Rare ইডেন্ট মানে আপনার ডেটাসেট এ এইরকম ইডেন্ট কম থাকবে। আর এইরকম ইডেন্টের ডেটাসেট যত কম থাকবে প্রেডিকশন ও ততটাই খারাপ আসবে। তবে এটা নিয়ে চিন্তা না করাই ভাল। আগে গতানুগতিক প্রেডিকশন ঠিক করেন, পরে না হ্য রেয়ার ইডেন্ট ঠিক করলেন।

আরও কিছু অ্যানালাইসিস।

True / False Ratio চেক করা

আমরা চাইলে দেখতে পারি, এই ডেটাসেট এ শতকরা কতজন ডায়াবেটিসে আক্রান্ত আৰ কতজন নয়, নোটবুক বেৱ
কৰে ঝটপট কোড লিখে ফেলেন।

```
num_true = 0.0
num_false = 0.0
for item in data_frame['diabetes']:
    if item == True:
        num_true += 1
    else:
        num_false += 1

percent_true = (num_true / (num_true + num_false)) * 100
percent_false = (num_false / (num_true + num_false)) * 100

print "Number of True Cases: {0} ({1:2.2f}%)".format(num_true, percent_true)
print "Number of False Cases: {0} ({1:2.2f}%)".format(num_false, percent_false)
```

আউটপুট:

```
Number of True Cases: 268.0 (34.90%)
Number of False Cases: 500.0 (65.10%)
```

আমরা Pythonic Way তে কোডটা আসলে চার লাইনে লিখতে পারি।

```
# Pythonic Way
num_true = len(data_frame.loc[data_frame['diabetes'] == True])
num_false = len(data_frame.loc[data_frame['diabetes'] == False])
print "Number of True Cases: {0} ({1:2.2f}%)".format(num_true, ((float)num_true / (num_true + num_false)) * 100)
print "Number of False Cases: {0} ({1:2.2f}%)".format(num_false, ((float)num_true / (num_true + num_false)) * 100)
```

Data Rule #4

ডেটা ম্যানিপুলেশন হিস্ট্রি রাখবেন ও চেক কৱবেন নিয়মিত

- এটা কৰাৰ জন্য একটা ব্যবস্থা আছেই (Jupyter Notebook ব্যবহাৰ কৰে)
- ভাৰ্সন কন্ট্ৰোল সিস্টেম ব্যবহাৰ কৰে, যেমন : Git, SVN, BitBucket, GitHub, GitLab ইত্যাদি

সামারি

কী কী কৱলাম এই দুই পৰ্বে?

- Pandas দিয়ে ডেটা রিড কৱলাম

- কো-রিলেশন সম্পর্কে ধারণা নিলাম
- ডুপ্লিকেট কলাম উচ্ছেদ করলাম
- ডেটা মোন্ট করলাম
- True/False রেশিও চেক করলাম

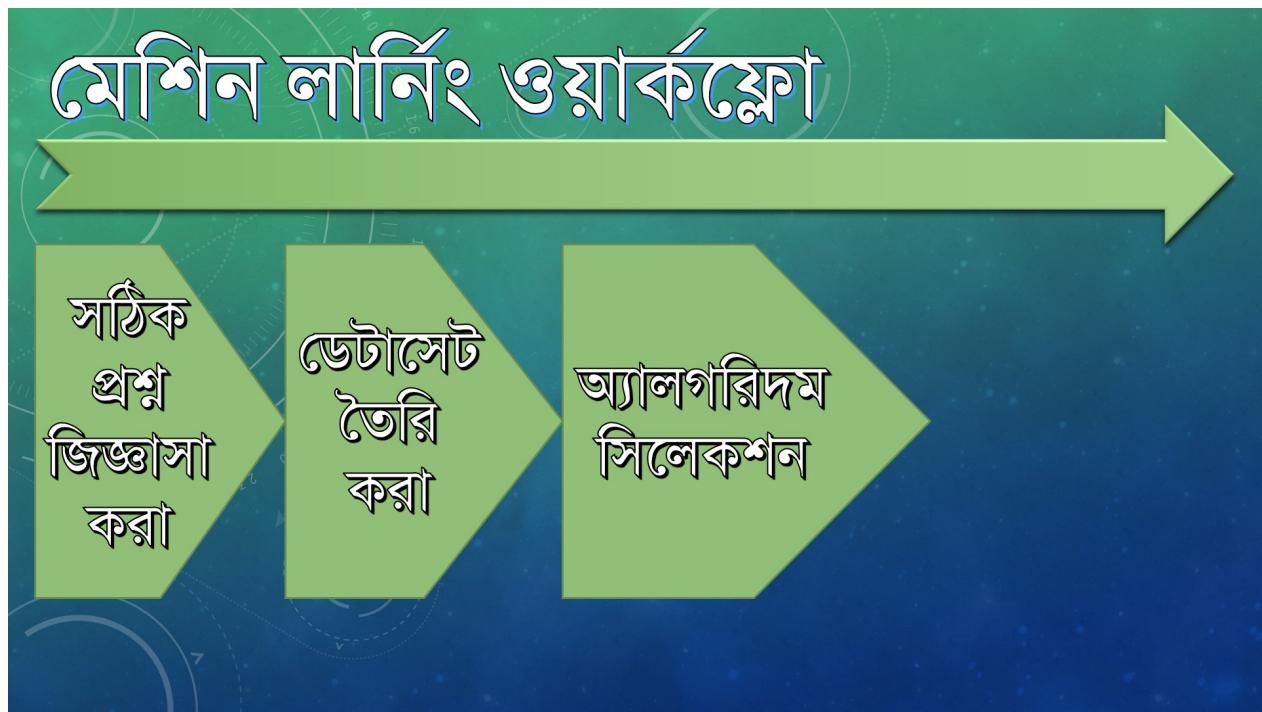
So far so good, পরবর্তী পর্বে আশা করি আমরা অ্যালগরিদম অ্যাপ্লাই করে প্রে�িক্ষন করা শুরু করে দিব।

Prediction is very difficult, especially if it's about the future - Niels Bohr

অ্যালগরিদম সিলেকশন

দেখতে দেখতে আমরা মেশিন লার্নিংয়ের তৃতীয় ধাপে এসে পড়লাম। এত এত লার্নিং অ্যালগরিদমের মধ্যে কোনটা আমার জন্য বেস্ট চয়েস হবে সেটা কীভাবে নির্ধারণ করব।

কাজের ধারা যদি আবেকবাব দেখি তাহলে এমন দাঁড়াবে,



এই চ্যাপ্টারের ওভারভিউ

আমরা এই চ্যাপ্টারে পর্বে আলোচনা করব,

- লার্নিং অ্যালগরিদমের কাজ কী
- অ্যালগরিদম সিলেক্ট করব কোন কোন ক্রাইটেরিয়ার ভিত্তিতে
 - অ্যালগরিদম বাছাইয়ের জন্য সল্যুশন স্টেটমেন্ট ব্যবহার করব
 - বেস্ট অ্যালগরিদম কোনটা হবে সেটা নিয়ে আলোচনা করব
 - প্রাথমিক অ্যালগরিদম বাছাই করব
 - প্রাথমিক বলার কারণ হচ্ছে, একই প্রবলেম অনেক সময় একটা অ্যালগরিদম দিয়ে রান করা ঠিক না, সবসময় আমাদের বেস্ট অ্যালগরিদমের পেছনে ছুটতে হবে। তাই যতটা পারা যায় একই ডেটাসেট বিভিন্ন অ্যালগরিদম দিয়ে ট্রেইন করে পার্ফর্মেন্স টেস্ট করা জরুরি। তবে কাজ করার জন্য প্রথমে একটা অ্যালগরিদম নিয়ে বাছাই করতে হবে, সেটাই এখানে আলোচনা করা হবে।

লার্নিং অ্যালগরিদমের কাজ

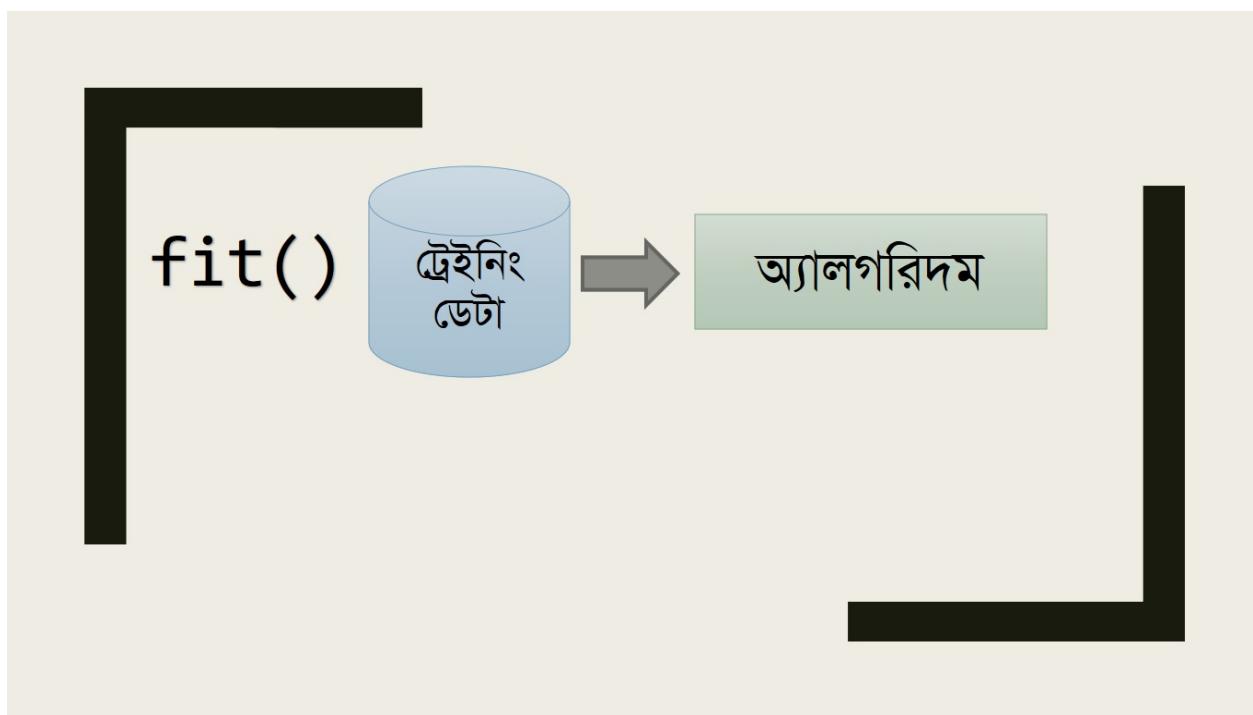
ব্যাপারটা হস্যকর শোনালেও প্রথমে আমাদের বুঝতে হবে মেশিন লার্নিং প্রসেসে অ্যালগরিদমের কাজ কী। তাহলে একটু দেখা যাক,

লার্নিং অ্যালগরিদমকে ইঞ্জিনের সাথে তুলনা করা যায় যেটা পুরো মেশিন লার্নিং প্রসেস পরিচালনা করে। ডেটা প্রিপ্রেসিং পরের গুরুত্বপূর্ণ কাজই হল লার্নিং অ্যালগরিদমের কাজ।

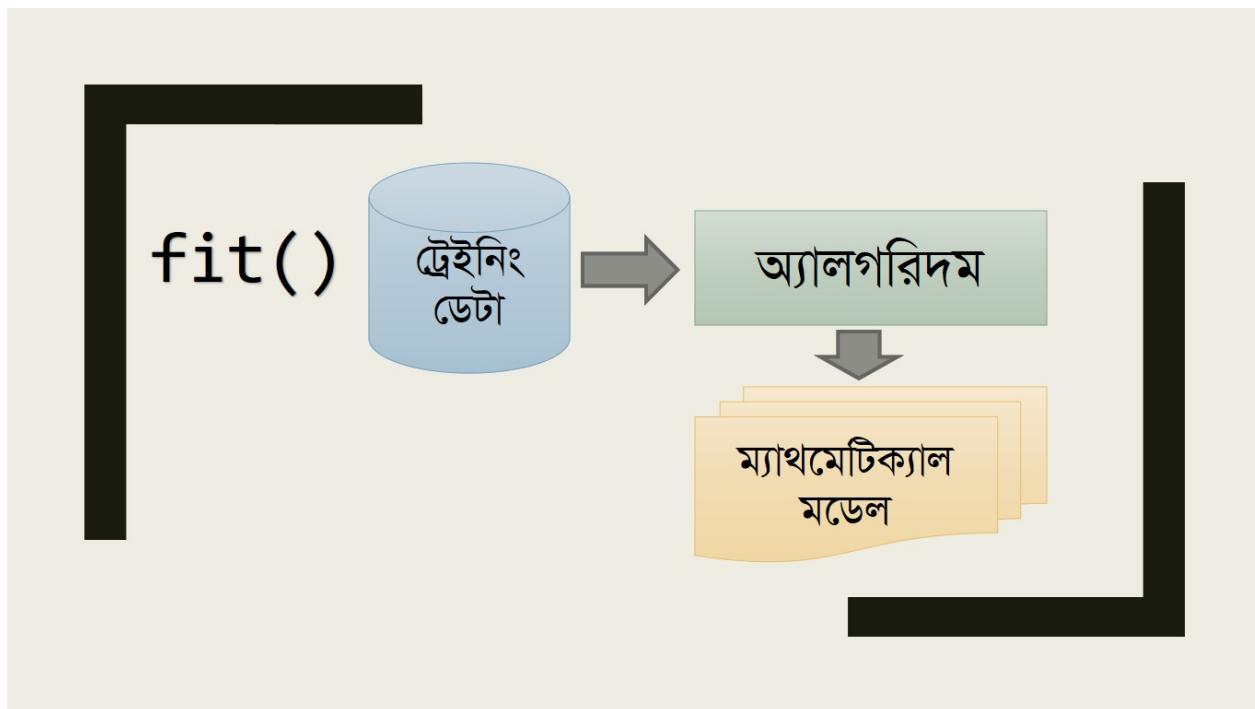
আমরা প্রথমে আমাদের ডেটাসেটকে দুইভাবে ভাগ করি,

- ট্রেইনিং ডেটা (বেশি পরিমাণে থাকে; টেস্টিং ডেটা এখান থেকে বাদ দেওয়া হয়)
- টেস্টিং ডেটা (অল্প পরিমাণে থাকে; ট্রেইনিং ডেটাসেট এর কোন ডেটা টেস্টিং ডেটাসেট এ থাকে না)

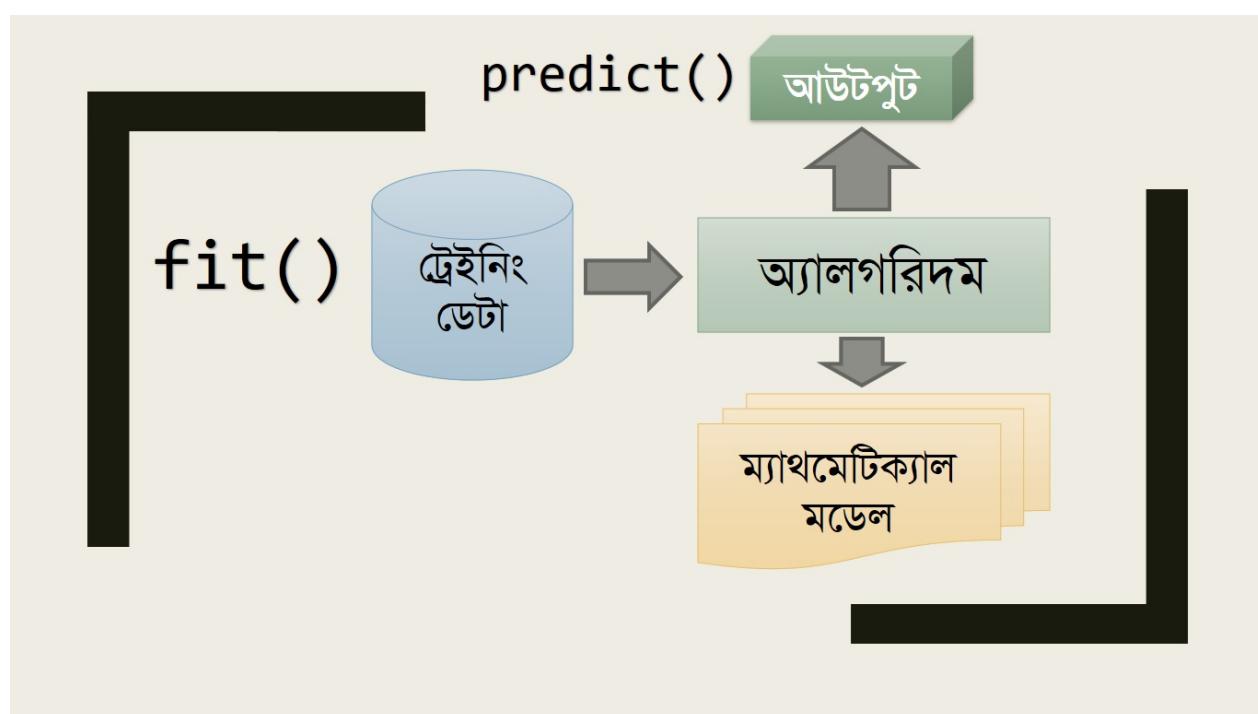
এবার এই ট্রেইনিং ডেটা আমরা অ্যালগরিদমে Feed করি, সাধারণত `Scikit-learn` এ অ্যালগরিদমে ফিড ও অ্যানালাইসিসের কাজ করার জন্য `fit()` ফাংশন ব্যবহার করলেই হয়।



এই অ্যালগরিদমের পিছনে কাজ করে ম্যাথমেটিক্যাল মডেল। এই ম্যাথমেটিক্যাল মডেলের মাধ্যমে অ্যালগরিদম ডেটাসেট অ্যানালাইসিসের সময় ইটারনাল প্যারামিটারগুলো ঠিকঠাক করে নেয়। এই কাজগুলো বোঝার জন্য ম্যাথ নিয়ে আলোচনা করা দরকার, কিন্তু আমরা আপাতত কাজ চালিয়ে নেওয়ার জন্য সবকিছু `magic` হিসেবে বিবেচনা করে কাজ আগাতে পারি। অবশ্যই আমরা ম্যাথমেটিক্যাল অ্যানালাইসিস দেখব, কিন্তু এখন পার্ফেক্ট সময় নয়। ম্যাথ নিয়ে গুরুত্বপূর্ণ করলে হ্যত ইটারেন্স হারাতে পারেন, তাই আমরা আগে গাড়ি চালানো শিখব পরে দেখব ইঞ্জিন অর্থাৎ, লার্নিং অ্যালগরিদম কীভাবে কাজ করে।



এর পরের কাজ সহজ, `predict()` ফাংশন কল করার মাধ্যমে আমরা ডেটাসেট এ নাই এমন জিনিস প্রেডিষ্ট করতে পারি (যেমন, ডায়াবেটিস নির্ণয়ের জন্য আগে Pima Indian Dataset দিয়ে মডেল ট্রেইন করব `fit()` ফাংশন দিয়ে, তারপর যেকোন ব্যক্তির ও প্যারামিটারের ডেটাগুলো দিয়ে `predict()` ফাংশনের মাধ্যমে জানব তার ডায়াবেটিস হওয়ার সম্ভাবনা করত্তুকু)



কোন অ্যালগরিদম দিয়ে ট্রেইন ও প্রেডিষ্ট করব?

প্রায় ৫০ টার উপরে প্রতিষ্ঠিত লার্নিং অ্যালগরিদম আছে। আবার এগুলোর মাঝে ক্রসওভার করিয়ে আপনি নিজেও কাস্টম অ্যালগরিদম তৈরি করতে পারেন। কিন্তু কীভাবে বুঝব আমাদের কাজের জন্য কোন অ্যালগরিদমটা পার্ফেক্ট? সেটা নিয়ে আলোচনা করার জন্যই এই টপিক।

অ্যালগরিদম বাছাই করার জন্য প্রত্যেকের নিজস্ব কিছু সিলেক্ট করা ফ্যাট্টের থাকে। যখন আপনি এক্সপার্ট হবেন তখন আপনি নিজেই বুঝতে পারবেন কোন অ্যালগরিদম কোন কাজের জন্য বেষ্ট।

আপাতত এই ফ্যাট্টেরগুলোর উপর ভিত্তি করে বাছাই করতে পারেন।

Algorithm Decision Factors

- লার্নিং টাইপ (Supervised নাকি Unsupervised)
- ৱেজাল্ট (Value নাকি Yes/No টাইপ উত্তর)
- কম্প্লেক্সিটি (Simple নাকি Complex)
- বেসিক না অ্যাডভান্সড

আমরা সল্যুশন স্টেটমেন্ট ও ওয়ার্কফ্লো প্রসেস, দুইটার কম্বিনেশনে সিলেক্ট করব কোন অ্যালগরিদম বাছাই করা ভাল হবে।

• লার্নিং টাইপ

একেক অ্যালগরিদমের লার্নিং প্রসেস আলাদা। চলুন সল্যুশন স্টেটমেন্ট আগে একবার দেখি তারপর ঠিক করি আমাদের কোন ধরণের লার্নিং দরকার।

মেশিন লার্নিং ওয়ার্কফ্লো ব্যবহার করে Pima Indian Data কে প্রিপ্রসেস ও প্রয়োজনীয় ট্রান্সফর্মেশন করার পর একটা প্রেডিস্টিভ মডেল তৈরি করতে হবে। এবার এই মডেলকে ৭০% বা তারও বেশি অ্যাকুরেসির সাথে নির্ণয় করতে হবে কে কে ডায়াবেটিসে আক্রত হতে পারে।

উপরের স্টেটমেন্টে বোল্ড করা অংশতে আমরা দেখতে পাই সেখানে প্রেডিস্টিভ মডেল বিল্ড করার কথা বলা হয়েছে।

আমরা জানি,

Prediction Model => Supervised Machine Learning

অর্থাৎ, আমরা পেয়ে গেলাম আমাদের বাছাইকৃত অ্যালগরিদমের লার্নিং টাইপ কী হবে।

ফাইনালি, যেসব অ্যালগরিদম **Unsupervised Learning** নিয়ে কাজ করে ওগুলো আমরা ব্যবহার করব না

এতে করে,

২২ টি অ্যালগরিদম বাদ পড়ে গেল, হাতে থাকল ২৮ টা অ্যালগরিদম

তাও অনেক! সমস্যা নাই, একে একে আমরা আরও ফিল্টার করতে পারব।

• ৱেজাল্ট টাইপ

আগেই বলেছি, আমরা সাধারণত দুই ধরণের উভয় পেতে চাই। একটা হল ভ্যালু (Regression: যেমন বাড়ির আকারের সাথে দরদাম কেমন হবে) আরেকটা হল হ্যাঁ/না টাইপ উভয় (Classification)।

এখনে বোঝাই যাচ্ছে ডায়াবেচিস সমস্যা নির্ধারণ আসলে Classification, কারণ আমরা জানতে চাচ্ছি ডায়াবেচিস হবে কি না। আবার, ডিস্ক্রিট ভ্যালু; যেমন: 1-100, 101-200, 201-300 বা small, medium, large ইত্যাদিও ক্লাসিফিকেশনের মধ্যে পড়ে।

এবার দেখি কয়টা অ্যালগোরিদম বাদ পড়ল,

বেজাল্ট টাইপ ক্লাসিফিকেশন হওয়াতে বাদ পড়ল ৮ টা অ্যালগোরিদম, হাতে রইল ২০ টা

• কমপ্লেক্সিটি

আমরা যেহেতু মেশিন লার্নিং শেখা শুরু করেছি সেহেতু আমাদের উচিত জটিল অ্যালগোরিদমগুলো এড়িয়ে চলা।
তারমানে KISS (Keep It Short and Simple) ফরমুলা অ্যাপ্লাই করা।

জটিল অ্যালগোরিদম কোনগুলো?

• Ensemble Algorithms:

- এগুলো স্পেশাল অ্যালগোরিদম, কারণ একেকটা Ensemble Algorithm অনেকগুলা অ্যালগোরিদমের সমষ্টি হয়ে থাকে।
- খুবই ভাল পার্ফর্মেন্স
- ডিবাগিং করা সুবিধাজনক নয়

এতে করে অ্যালগোরিদম কমে দাঁড়াল ১৪ টাতে

• Basic নাকি Enhanced?

Enhanced

- Basic এর ভ্যারিয়েশন
- পার্ফর্মেন্স Basic এর চেয়ে বেটার (বলা লাগে নাকি? :P)
- অতিরিক্ত সুবিধাসমূহ
- Complex

Basic

- সহজ
- তাই সহজে বোঝা যায়

হ্যাঁ বুঝতে পারছেন, যেহেতু আমরা বিগিনার, তাই আমাদের Basic এ থাকাই ভাল।

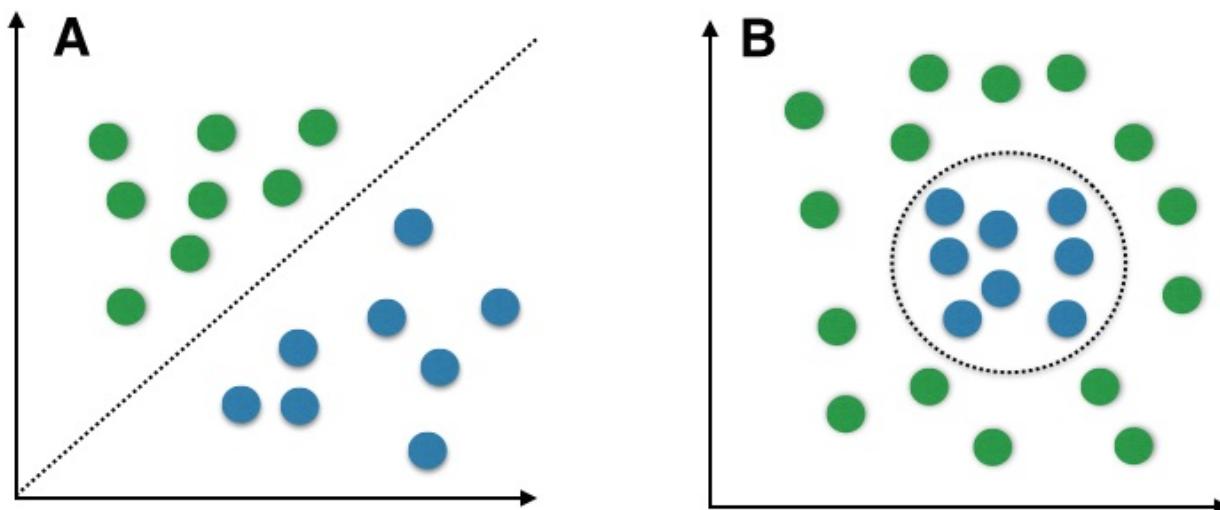
ফিল্টারিং শেষে তিনটা Candidate Algorithm

আমরা এখন তিনটা অ্যালগরিদম,

- Naive Bayes
- Logistic Regression
- Decision Tree

এর মধ্য থেকে একটা বাছাই করব। তিনটা সম্পর্কে আগে অন্ন কিছু আলোচনা করার পর আমরা একটা ডিসিশনে আসব যে কোনটা ব্যবহার করা বেটার। তিনটাই মেশিন লার্নিংয়ের বেসিক ও স্লাসিক অ্যালগরিদম। জটিল অ্যালগরিদমগুলো মূলত এগুলোকে বিস্তৃত হিসেবে ব্যবহার করে গঠিত। Naive Bayes দিয়ে শুরু করা যাক।

• Naive Bayes



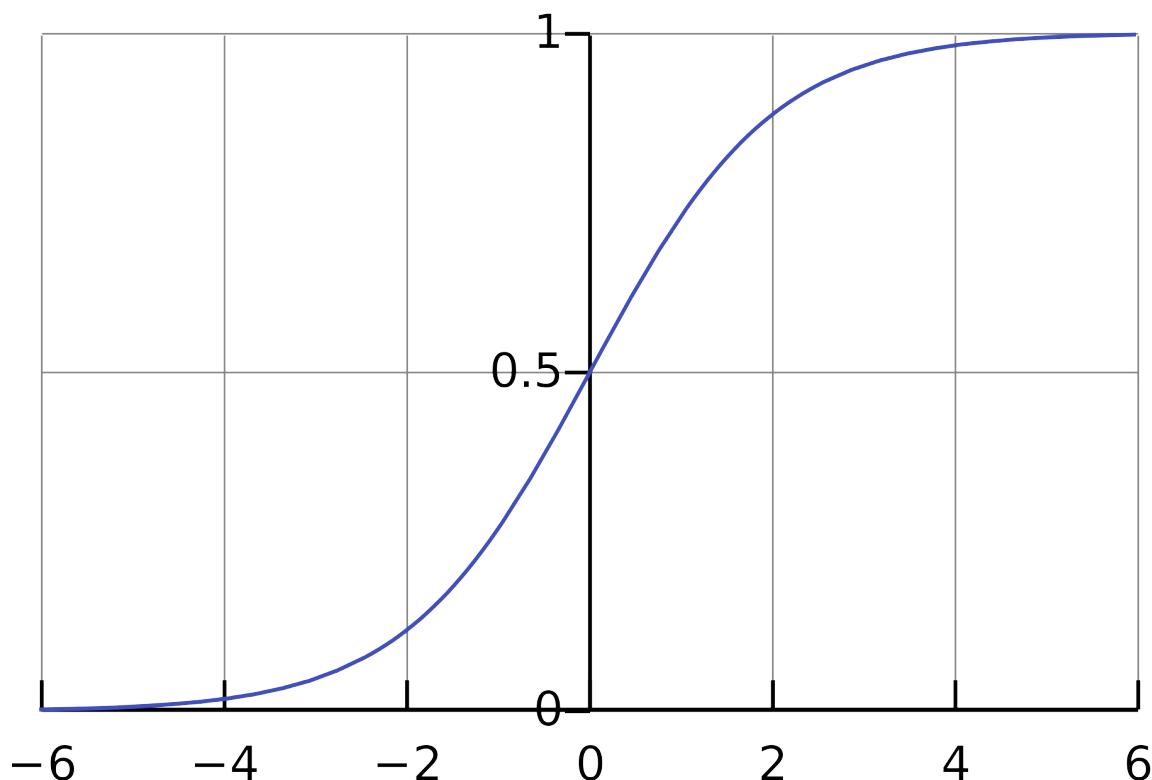
Naive Bayes অ্যালগরিদম 'Bayes Theorem' অ্যাপ্লাই করে বানানো। যারা 'Bayes Theorem' এর নাম শোনেন নি তাদের জন্য বলা যেতে পারে, Probability এর অন্যতম ফার্মান্টাল থিওরেম হল 'Bayes Probability Theorem'। খুবই গুরুত্বপূর্ণ থিওরেম হওয়াতে এটা নিয়ে বিস্তারিত পরে একসময় আলোচনা করা হবে। (আবারও ম্যাথেমেটিক্স)

'Naive Bayes' অ্যালগরিদম কোন কিছু হওয়ার সম্ভাবনা নির্ধারণ করে। যেমন, High Blood Pressure এর সাথে ডায়াবেটিস হওয়ার সম্ভাবনা কেমন? এভাবে বিভিন্ন 'Feature / Input Variable' এর সাথে Probability মিল্ক করে কোন ঘটনা হওয়ার সম্ভাবনা নির্ধারণ করে এই অ্যালগরিদম (অবশ্যই পূর্বের ডেটাসেট এর উপর ডিপ্তি করে)।

এর কিছু বৈশিষ্ট্য হল

- ঘটনা ঘটার সম্ভাবনা নির্ধারণ করে
- প্রতিটি ফিচার বা ইনপুট ড্যারিয়েবল (আলোচ্য সমস্যার ক্ষেত্রে: no of preg, insulin, ইত্যাদি) সমান গুরুত্বপূর্ণ।
 - তারমানে এখানে Blood Pressure আর BMI (Body Mass Index) সমান গুরুত্বপূর্ণ (পাশাপাশি সব ড্যারিয়েবল-ই)
- প্রেডিকশনের জন্য অন্ন পরিমাণ ডেটাই যথেষ্ট

- লজিস্টিক রিগ্রেশন (Logistic Regression)



নামটা কনফিউজিং, মনে আমরা জানতাম Regression মানে Continuous সম্পর্কিত। কিন্তু Classification হল Discrete ভ্যালু। মনে হতেই পারে, Classification করার জন্য আমরা Regression মেথড নিয়ে আলোচনা কেন করছি?

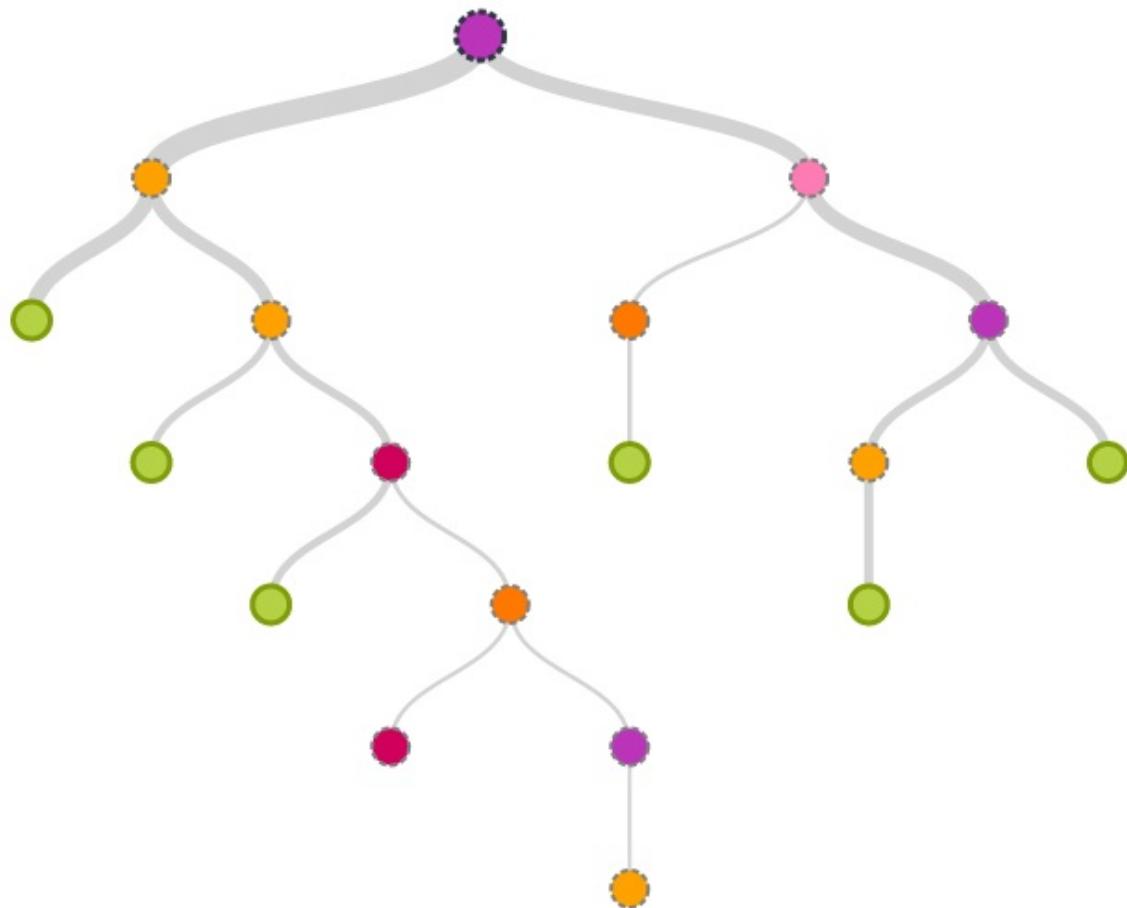
আসলে Logistic Regression এর আউটপুট ১ (.৯৯৯৯) বা ০ (০.০০০০১) হয়।

বৈশিষ্ট্য

- বাইনারি রেজাল্ট
- Input Variable / Feature এর রিলেশনশিপ Weighted করা হয় (সবগুলা ফিচার সমান গুরুত্বপূর্ণ নাও হতে পারে)

পরবর্তী অ্যালগরিদম দেখা যাক।

- ডিসিশন ট্রি (Decision Tree)



- এর গঠন বাইনারি ট্রি এর মত (ডেটা স্ট্রাকচার পড়ে থাকলে ধারণা করতে পারবেন)
- প্রতিটা নোড আসলে একেকটা ডিসিশন
- প্রচুর পরিমাণ ডেটা লাগে ডিসিশন স্প্লিটিংয়ের জন্য

অবশেষে সিলেক্ট করলাম **Naive Bayes**

কেন?

- সহজে বোধ্য যায়
- দ্রুত কাজ করে (প্রায় ১০০ গুণ দ্রুত সাধারণ অ্যালগরিদমের তুলনায়)
- ডেটা চেঞ্জ হলেও মডেল স্টেবল থাকে
 - ডিবাগিং করা তুলনামূলক সহজ
- সবচেয়ে বড় কারণ হল, আমাদের সমস্যার সাথে এই অ্যালগরিদমটা পুরোপুরি ম্যাচ করে, কারণ আমরা likelihood বের করতে চাচ্ছি এবং এই অ্যালগরিদমের কাজই হল likelihood নির্ণয় করা :)

সারাংশ

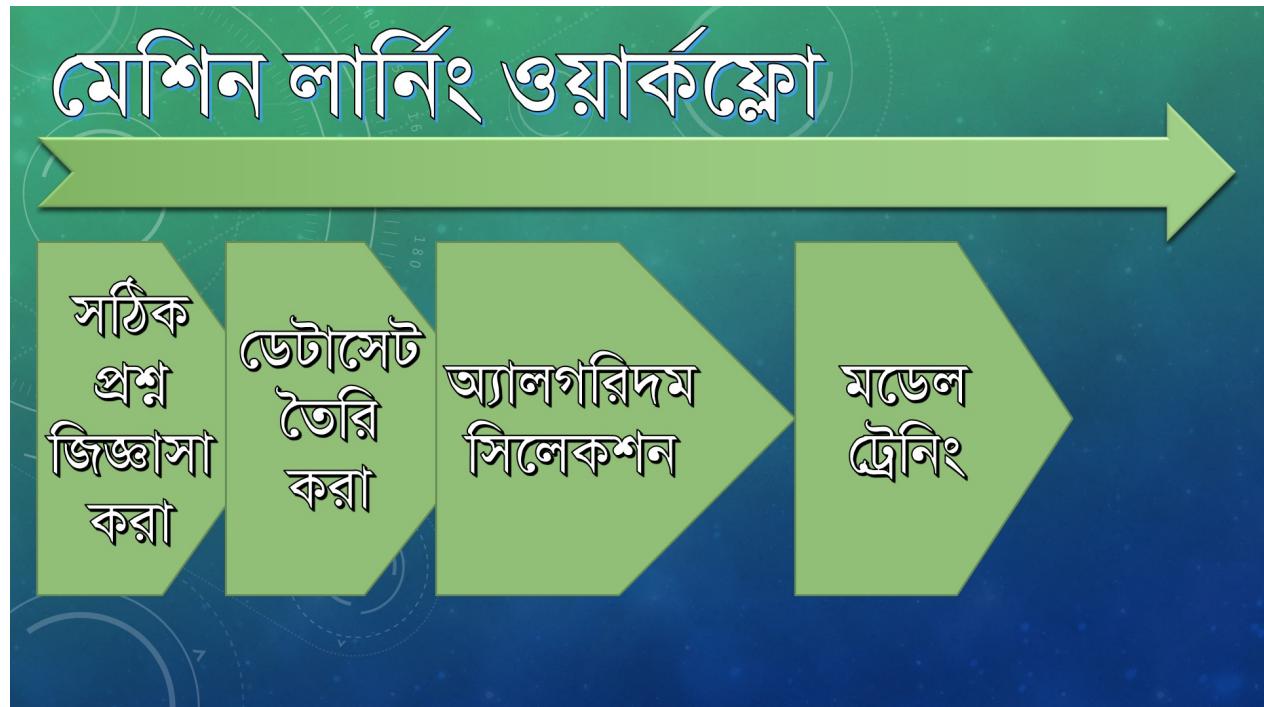
- প্রচুর লার্নিং অ্যালগরিদম অ্যাডেইলেবল
- সিলেকশন করলাম

- Learning Type - Supervised
- Result - Binary Classification
- Complexity - Non-Ensemble
- Basic or Enhanced - Basic
- Naive Bayes সিলেষ্ট কৱলাম ট্ৰেইনিংয়ের জন্য, কাৰণ
 - সহজ, ফাস্ট ও স্টেবল

পৰবতী চ্যাপ্টাৰেই আমৰা আশা কৰি প্ৰেডিষ্ট কৰতে পাৰব (Promise!)

মডেল ট্রেইনিং

আমরা এই পর্ষ্ণ সল্যুশন স্টেটমেন্ট, ডেটা কালেকশন ও প্রিপসেসিং এবং অ্যালগরিদম সিলেকশন পর্ষ্ণ কাজ করেছি। এখন আমরা দখল মডেল ট্রেইন করতে হয় কীভাবে। ফ্লোচার্ট অনুযায়ী আমরা নিচের ধাপে আছি,



এই চ্যাপ্টারের ওভারভিউ

- ট্রেইনিং প্রসেস সম্পর্কে আলোচনা করা
- `Scikit-learn` প্যাকেজ সম্পর্কে আরো ধারণালাভ করা
- অ্যালগরিদম ডায়াবেটিস ডেটা দিয়ে ট্রেইন করে ট্রেইনড মডেল তৈরি করা

মেশিন লার্নিং ট্রেইনিং

সংজ্ঞানুযায়ী,

Letting specific data teach a Machine Learning Algorithm to create specific prediction model.

তার মানে, যে ট্রেইনিংয়ের মাধ্যমে একটা মেশিন লার্নিং মডেলকে স্পেসিফিক ডেটাসেট দিয়ে ট্রেইন করে একটা স্পেসিফিক প্রেডিকশন মডেলে পরিণত করা যায় সেটাই মেশিন লার্নিং ট্রেইনিং।

স্বাভাবিক, আমি ডায়াবেটিসের ডেটাসেট দিয়ে মডেল ট্রেইন করে রোদ-বৃষ্টি হওয়া প্রেডিক্ষন করতে পারব না। তার জন্য আমাদের আলাদা স্পেসিফিক ডেটাসেট লাগবে।

প্রায়ই মডেল রিট্রেইন (retrain) করার প্রয়োজন হয়।

কেন আমরা মডেল **retrain** করি?

- ধরি Pima Indian dataset কিছুদিন পর পর আপডেটেড হয় নতুন নতুন ডেটা দিয়ে (মানে অবজারভেশন অর্থাৎ নতুন নতুন Row অ্যাড হচ্ছে)। আমরা মেশিন লার্নিংয়ের বৈশিষ্ট্য অনুযায়ী জানি, যত ডেটাসেট থাকবে তত বেটার, তাই নতুন অ্যাড হওয়া ডেটাসেট দিয়ে আবার মডেল ট্রেইন করলে আমরা অবশ্যই আগের চেয়ে ভাল ফলাফল পাব।

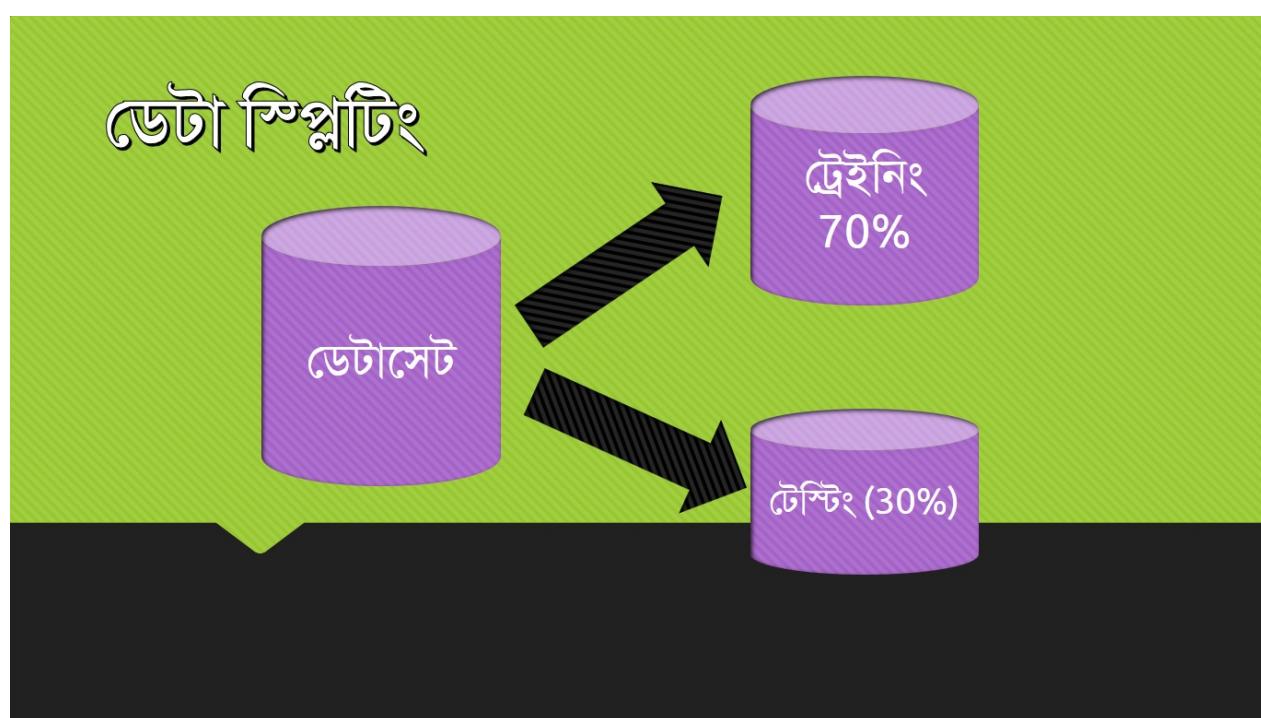
'New Data => better predictions'

- নতুন ডেটাসেট থেকে আবারও আমরা কিছু ডেটা ট্রেইনিং আর কিছু ডেটা টেস্টিংয়ের জন্য রেখে পারফর্মেন্স ডেরিফাই করতে পারি।

ট্রেইনিং ও ডারভিউ

ডেটাসেট স্প্লিটিং

ট্রেইনিংয়ের শুরুতেই যেটা করতে হবে সেটা হল ডেটাসেট ভাগ করে নিতে হবে। গড়পড়তায় আমরা সাধারণত 70% ডেটা রাখি ট্রেইনিংয়ের জন্য এবং বাকি 30% রাখি টেস্টিংয়ের জন্য।



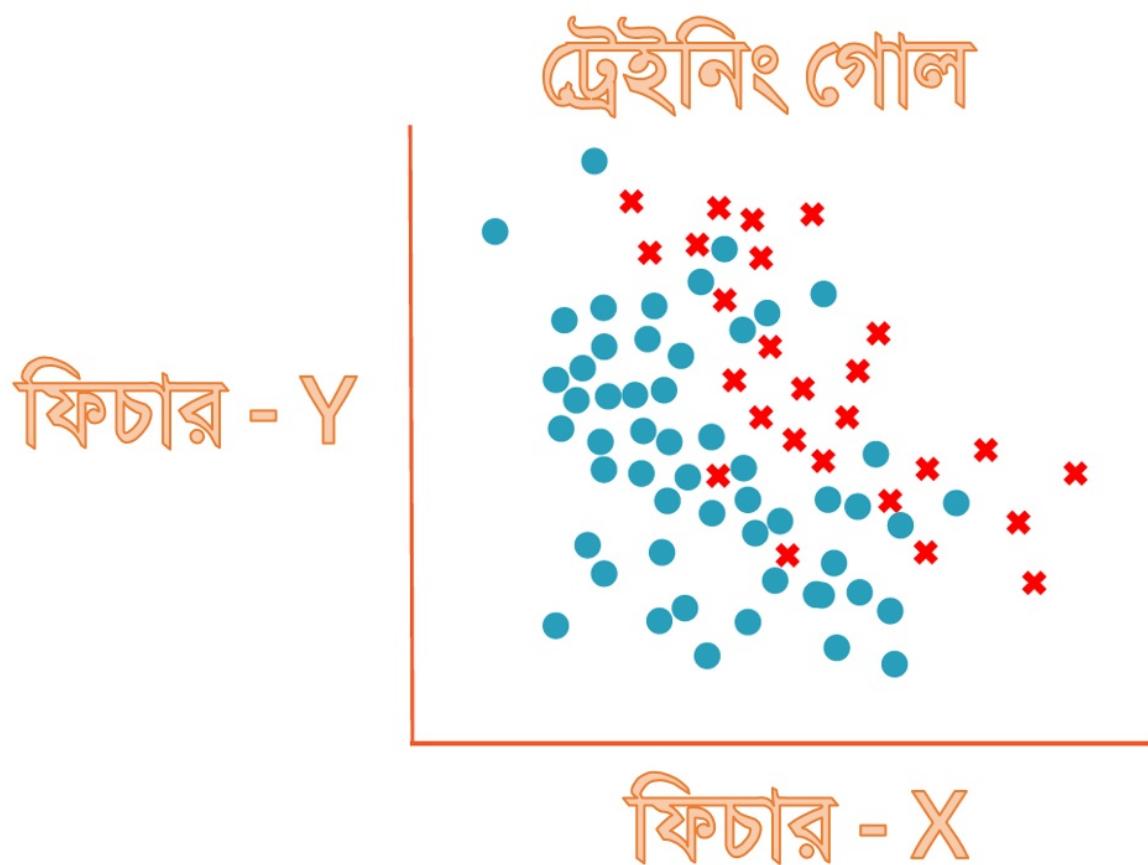
ট্রেইনিং ডেটা অ্যালগরিদমে ফিল্ড করার মাধ্যমে আমরা অ্যালগরিদম ট্রেইন করি। কোন অ্যালগরিদম ট্রেইন করার আসল অর্থ হল ও স্পেসিফিক ডেটাসেট এর জন্য অ্যালগরিদমের ইন্টারনাল প্যারামিটারগুলো সেট করা। আমরা যখন ম্যাথমেটিক্যাল অ্যানালাইসিস দেখব তখন বিষয়টা ক্ষিয়ার হবে।

আগে দেখি আমাদের ট্রেইনিং গোল (Training Goal) আসলে কী?

ট্রেইনিং গোল ও ট্রেইনিং ডেটা

ট্রেইনিং গোল বোঝার জন্য আমরা হাইপোথেটিক্যাল ডেটাসেট নিচ্ছি। আবারও বলছি "ট্রেইনিং গোল" বোঝার জন্য আমরা আপাতত Diabetes ডেটাসেট ব্যবহার করছি না।

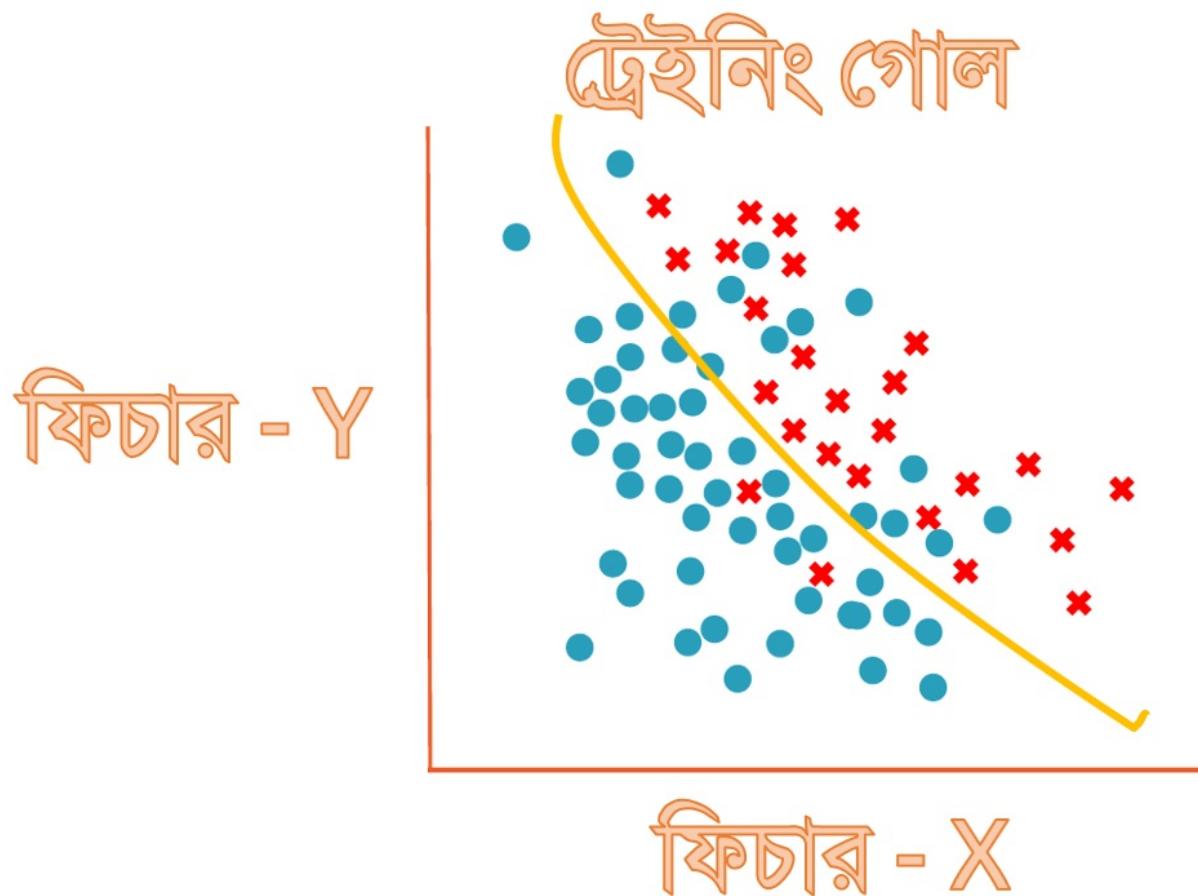
ধরি, সর্দি হবে কী না, সেটা বোঝার জন্য দুইটা ইনপুট ড্যারিয়েবল / ফিচার যথেষ্ট। ফিচার দুটো হল X & Y । প্রতি X এর সাপেক্ষে আমরা যদি Y এর Scatter প্লট করি,



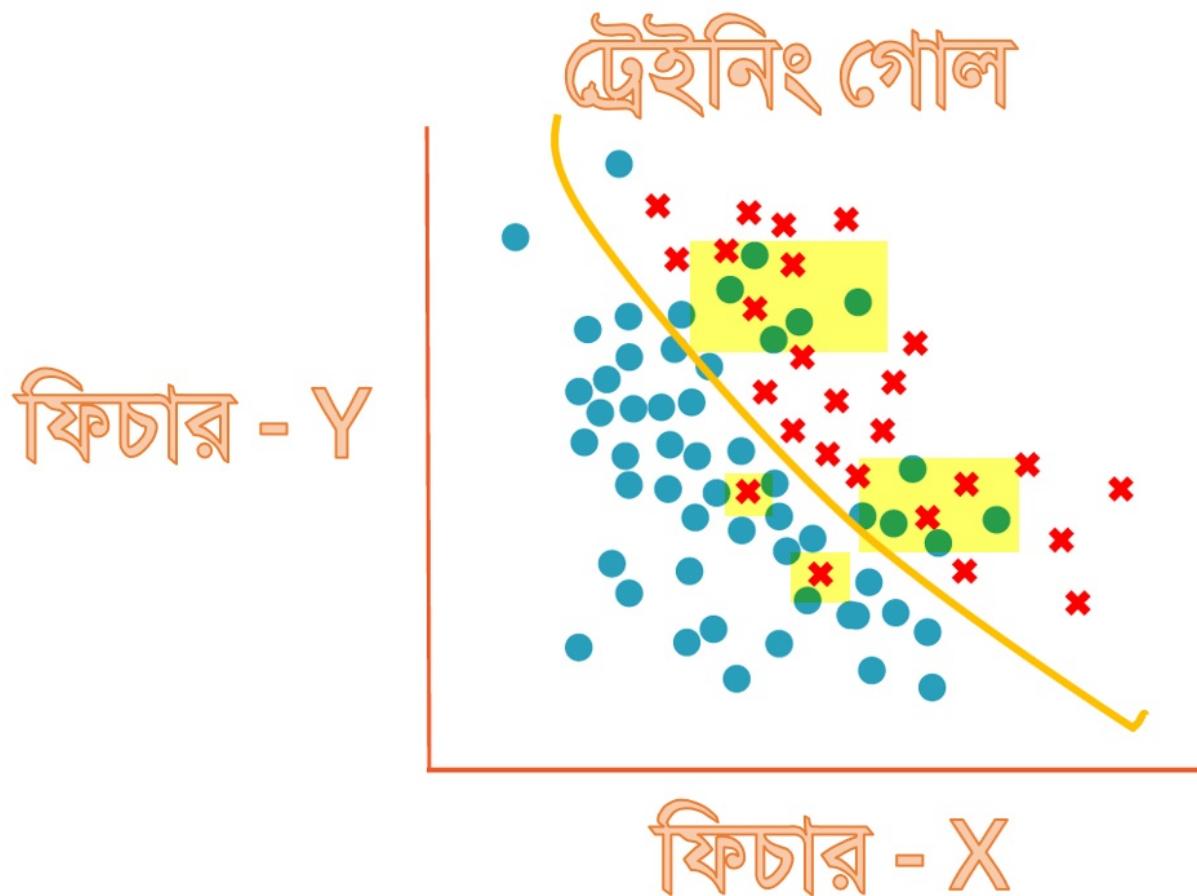
স্ক্যাটার প্লটটির ব্যাখ্যা:

এখানে নীল রংয়ের ডট গুলো দ্বারা বোঝানো হচ্ছে X ও Y এর ও কম্বিনেশনের জন্য সর্দি হবে না এবং লাল রংয়ের ডট দিয়ে বোঝানো হচ্ছে X ও Y এর ও সকল কম্বিনেশনের জন্য সর্দি হবে।

আমরা একটা সাধারণ ডিসিশন বাট্টারি ড্র করতে পারি ডটগুলো আলাদা করার জন্য। এটা মূলত করে দেবে আপনার ট্রেইনড অ্যালগরিদম। উপরের ডেটাসেটের জন্য এইরকম একটি ডিসিশন বাট্টারি এঁকে দিতে পারে আপনার অ্যালগরিদম।



কিন্তু বাটনারি ড্র করার পর দেখা যাচ্ছে নীল অংশে বেশ কিছু লাল ক্রস চলে এসেছে এবং লাল অংশে বেশ কিছু নীল বৃত্ত চলে এসেছে।



তারমানে ট্রেইনিং 100% নয়। কিন্তু 100% অ্যাকুরেসি আমাদের লক্ষ্যও নয়। এত বেশি অ্যাকুরেট মডেল তৈরি করতে গেলে Overfitting হওয়ার চান খুবই বেশি থাকে। Overfitting এবং Underfitting মেশিন লার্নিংয়ের গুরুত্বপূর্ণ বিষয়, তাই বরাবরের মত এগুলো নিয়েও আমরা বিস্তারিত আলোচনা পরে একসময় করে নেব।

এই ট্রেইনিং ডেটা ব্যবহার করে মডেলকে ট্রেইন করে আমরা ডিসিশন বাউন্ডারি তৈরি করিয়ে নিতে পারি। আপাতত এটাই হচ্ছে আমার ট্রেইনিং ডেটা ব্যবহারের উদ্দেশ্য।

টেস্টিং ডেটার কাজ কী?

মনে হওয়া স্বাভাবিক, যে 100% ডেটা আমরা কেন ট্রেইনিংয়ে কাজে লাগাচ্ছি না? কেন আমরা 70-30% এ স্পিল্ট করছি?! এতে করে ট্রেইনিং ডেটা কমে গেল না? তাতে পর্ফর্মেন্স খারাপ হবে না? সব ডেটা দিয়ে ট্রেইন করলে সমস্যা কোথায়?

হ্যাঁ অনেকগুলো প্রশ্ন এবং আমার চেষ্টা থাকবে সবগুলোর জবাব দেওয়ার জন্য।

100% ডেটা আমরা কেন ট্রেইনিংয়ে কাজে লাগাচ্ছি না? কেন আমরা 70-30% এ স্পিল্ট করছি?! এতে করে ট্রেইনিং ডেটা কমে গেল না?

প্রশ্নগুলো আসলে একই, আমরা পূর্বের আলোচনা করা একটি উদাহরণের মাধ্যমে বিষয়টা বোঝার চেষ্টা করি।

ধৰি, আমি কাউকে (মনে কৰন সে গুণ কৰতে জানে না, শুধু যোগ কৰতে জানে) নামতা শেখাচ্ছি। লক্ষ্য কৰবেন আমি 'শেখাচ্ছি' কে বোল্ড হৰফে লিখেছি। তাৰমানে আমি তাকে নামতাৰ লজিক শেখাচ্ছি। এবাৰ যদি তাকে আমি,

$$\begin{aligned} 2 \times 1 &= 2 \\ 2 \times 2 &= 4 \\ 2 \times 3 &= 6 \\ 2 \times 4 &= 8 \\ 2 \times 5 &= 10 \\ 2 \times 6 &= 12 \\ 2 \times 7 &= 14 \\ 2 \times 8 &= 16 \\ 2 \times 9 &= 18 \\ 2 \times 10 &= 20 \end{aligned}$$

এই ২ এৰ নামতা বাবাৰ পড়িয়ে মুখ্য কৰালাম। আমি যদি এখন তাকে বলি, বলত $2 \times 3 = ?$ সে ঝটপট উত্তৰ দিতে পাৱে যে $2 \times 3 = 6$ । এখন আমি তাকে টেস্ট কৰাৰ জন্য যদি বলি আচ্ছা এবাৰ ৫ এৰ ঘৰেৰ নামতা বল। এবাৰ সে আৱ ঝটপট উত্তৰ দিতে পাৱে না, কোন কোন ক্ষেত্ৰে উত্তৰ দিতে নাও পাৰে।

এই উদাহৰণ দিয়ে যেটা বুঝালাম যে, আমি তাকে আদৌ কিছু শেখাতে পাৰি নাই। ১০০% ডেটা সাধাই দিয়ে আল্টিমেটলি আমি তাকে লজিক বেৱ কৰা থেকে বিৱত রাখলাম। কিন্তু আমি যদি এটা কৰতাম,

$$\begin{aligned} 2 \times 1 &= 2 \\ 2 \times 2 &= 4 \\ 2 \times 3 &= 6 \\ \\ 2 \times 5 &= 10 \\ 2 \times 6 &= 12 \\ 2 \times 7 &= 14 \\ 2 \times 8 &= 16 \\ \\ 2 \times 10 &= 20 \end{aligned}$$

এখনে দুইটা ডেটা মিসিং, এবং তাকে দায়িষ্ট দিলাম তুমি বেৱ কৰ মিসিং ভ্যালুগুলো কী হবে? সে এবাৱ লজিক বেৱ কৰাৰ চেষ্টা কৰবে। কোন মুখ্য বুলি আওড়াবে না। শুধু তাই না, আমি আসলে তাৱ উত্তৰ থেকে বেৱ কৰতে পাৱব সে আদৌ লজিক শিখতে পেৰেছে কিনা।

অর্থাৎ আমোৱা সব টেস্টিং ডেটা দিয়ে ডেরিফাই কৰতে পাৰি আমাৰ তৈরিকৃত মডেল আসলেই প্ৰেডিষ্টিভ কিনা, আমি জানি এমন কোন ডেটা আমাৰ কাছে আছে কিন্তু আমি ট্ৰেইনিংয়ে সেটা প্ৰোডাইড কৰি নাই। তাতে যদি মডেল কাছাকাছি প্ৰেডিষ্ট কৰতে পাৱে তাৰমানে আমি মোটামুটি সফল। কাৰণ আমি একটা মডেলকে ট্ৰেইনড কৰতে পেৰেছি। আমাৰ কাছে উত্তৰ নাই এমন প্ৰশ্ন জিজ্ঞাসা কৰে আমি কীভাৱে জানব আৱেকজন সঠিক উত্তৰ দিচ্ছে কীনা?

ইনপুট ভ্যারিয়েবল বা ফিচাৰ সিলেকশন

ফিচার সিলেকশন বা ফিচার ইঞ্জিনিয়ারিং আসলে ডেটা সায়েন্সের আলোচ্য বিশাল একটা টপিক। কারণ আগেই বলেছি, অনেক সময় ডেটাসেট এ থাকা বেশ কিছু ফিচার থাকে যেগুলো অদরকারী, সেটা বাদ দিলে প্রেডিকশন আরও ভাল হবে। এই অদরকারী ড্যারিয়েবল ছেঁটে দিয়ে দরকারী ফিচার সিলেকশনের ভাবিষ্ঠি নাম ফিচার ইঞ্জিনিয়ারিং।

আমরা ডেটাসেট ল্যানিংয়ের সময় ফিচার ইঞ্জিনিয়ারিং খাটিয়েছিলাম, সেটা হল কো-বিলেশন বের করে আমরা skin ছাঁটাই করি।

Pima Indian Diabetes ডেটাসেট এ সিলেক্টেড ফিচারগুলো:

- No of Pregnancies
- Glucose Concentration
- Blood Pressure
- Thickness
- Insulin
- Body Mass Index
- Diabetes Predisposition
- Age

Scikit-learn ব্যবহার করে মডেল ট্রেইনিং

অবশ্যে আমরা যিওরি পাঠ করে ব্যাপক জ্ঞানার্জনের পর বসলাম কোডিং করতে। প্রস্তুতি নিন, আমরা এখন মডেল ট্রেইন করা শুরু করব।

ট্রেইনিংয়ের শুরুতে কী করতে হবে মনে আছে? না থাকলে সমস্যা নাই,

ডেটা স্প্লিটিং

নিচের কোড এর মাধ্যমে আমরা 70-30% ডেটা স্প্লিট করব। 70% হল ট্রেইনিং ডেটা, বাকিটা টেস্টিং ডেটা।

জুপিটার নোটবুক বের করে আগের করা কোডটিতে লেখা শুরু করুন।

```

from sklearn.cross_validation import train_test_split

feature_column_names = ['num_preg', 'glucose_conc', 'diastolic_bp', 'thickness', 'insulin', 'bmi', 'diab_prod', 'age']

predicted_class_name = ['diabetes']

# Getting feature variable values

X = data_frame[feature_column_names].values
y = data_frame[predicted_class_name].values

# Saving 30% for testing
split_test_size = 30

# Splitting using scikit-learn train_test_split function

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = split_test_size,
random_state = 42)

```

random_state = 42 দেওয়ার মানে, প্রতিবার প্রোগ্রাম বাত করলে স্প্লিটিং যেন একই জায়গা থেকে হ্য সেটার গ্যারান্টি দেওয়ার জন্য।

ডেটাসেটের স্প্লিটিং কি আসলেই ৭০-৩০ হয়েছে? চেক করা যাক

Python 3

```

print("{0:0.2f}% in training set".format((len(X_train)/len(data_frame.index)) * 100))
print("{0:0.2f}% in test set".format((len(X_test)/len(data_frame.index)) * 100))

```

Python 2

```

print "{0:0.2f}% in training set".format((len(X_train)/len(data_frame.index)) * 100)
print "{0:0.2f}% in test set".format((len(X_test)/len(data_frame.index)) * 100)

```

আউটপুট:

```

69.92% in training set
30.08% in test set

```

কাছাকাছি!

কোন মিসিং ডেটা আছে কীনা? (0 ভ্যালু, null ভ্যালু নয়)

ଅନେକ ସମୟ ଏକଟା କଲାମେ ହୃତ ବିଭିନ୍ନ ଧରଣେର ମାନ ଆଛେ କିନ୍ତୁ ଆପଣି ଚେକ କରତେ ଗିଯେ ଦେଖିଲେନ ଅନେକଗୁଲା ଡ୍ୟାଲୁ 0 ଯେଟା ସଜ୍ଜବ ନୟ । ସେଟା ନିଯେ କୀଭାବେ ଡିଲ କରବେନ? ଏକଟା ଅୟାଲଗରିଦମ ଆଛେ ଯେଟା ଦିଯେ 0 ଡ୍ୟାଲୁଗୁଲୋ ରିଙ୍ଗେସ କରେ ଏକଟା ଗଡ଼ ଡ୍ୟାଲୁ ବସିଯେ କାଜ କରାର ମତ ଷେଟେ ନେଓଯା ଯାଏ, ସେଟା ଦେଖାର ଆଗେ ଚଲେନ ଦେଖି ଆମାଦେର କତଗୁଲା ଡ୍ୟାଲୁ ଆସଲେ 0!

Python 3

```
print("# rows in dataframe {}".format(len(data_frame)))
print("# rows missing glucose_conc: {}".format(len(data_frame.loc[data_frame['glucose
_conc'] == 0])))
print("# rows missing diastolic_bp: {}".format(len(data_frame.loc[data_frame['diastolic_bp'] == 0])))
print("# rows missing thickness: {}".format(len(data_frame.loc[data_frame['thickness'] == 0])))
print("# rows missing insulin: {}".format(len(data_frame.loc[data_frame['insulin'] == 0])))
print("# rows missing bmi: {}".format(len(data_frame.loc[data_frame['bmi'] == 0])))
print("# rows missing diab_pred: {}".format(len(data_frame.loc[data_frame['diab_pred'] == 0])))
print("# rows missing age: {}".format(len(data_frame.loc[data_frame['age'] == 0])))
```

Python 2

```
print "# rows in dataframe {}".format(len(data_frame))
print "# rows missing glucose_conc: {}".format(len(data_frame.loc[data_frame['glucose
_conc'] == 0]))
print "# rows missing diastolic_bp: {}".format(len(data_frame.loc[data_frame['diastolic_bp'] == 0]))
print "# rows missing thickness: {}".format(len(data_frame.loc[data_frame['thickness'] == 0]))
print "# rows missing insulin: {}".format(len(data_frame.loc[data_frame['insulin'] == 0]))
print "# rows missing bmi: {}".format(len(data_frame.loc[data_frame['bmi'] == 0]))
print "# rows missing diab_pred: {}".format(len(data_frame.loc[data_frame['diab_pred'] == 0]))
print "# rows missing age: {}".format(len(data_frame.loc[data_frame['age'] == 0]))
```

ଆଟ୍ରିବ୍ୟୁସନ୍:

```
# rows in dataframe 768
# rows missing glucose_conc: 5
# rows missing diastolic_bp: 35
# rows missing thickness: 227
# rows missing insulin: 374
# rows missing bmi: 11
# rows missing diab_pred: 0
# rows missing age: 0
```

- ০ ভ্যালু কে কোন ভ্যালু দিয়ে রিপ্লেস করার একটা টেকনিক হল **Imputation**। ইম্পুট করার জন্য সাইকিটে অলরেডি বেডিমেড কোড দেয়া আছে, আমরা আপত্ত সেটা ব্যবহার করে কাজ চালিয়ে নেব।

Python 2 & 3

```
from sklearn.preprocessing import Imputer

#Impute with mean all 0 readings
fill_0 = Imputer(missing_values=0, strategy="mean", axis=0)

X_train = fill_0.fit_transform(X_train)
X_test = fill_0.fit_transform(X_test)
```

এখানে `fill_0` ও কিন্তু একধরণের মডেল, যার কাজ হল 0 ভ্যালুগুলোকে `mean` স্ট্যাটেজির মাধ্যমে একটা লজিক্যাল ভ্যালু দিয়ে রিপ্লেস করা।

আমরা এই পরিবর্তিত ট্রেইন ভ্যালু দিয়ে ট্রেইন করব এবং টেস্ট ভ্যালু দিয়ে টেস্ট করব।

y_train বা **y_test** এ কেন **Imputer** ব্যবহার করলাম না?

কারণ সিম্পল, ওখানে কোন মিসিং ডেটা নাই।

মডেল ট্রেইনিং

অবশ্যে আমরা সেই ম্যাজিক্যাল ফাংশন কল করার মাধ্যমে মডেল ট্রেইন করব।

Python 2 & 3

```
from sklearn.naive_bayes import GaussianNB

# create Gaussian Naive Bayes model object and train it with the data
nb_model = GaussianNB()

nb_model.fit(X_train, y_train.ravel())
```

আমরা আগেই আলোচনা করে ঠিক করেছিলাম যে আমাদের সিলেক্টেড অ্যালগরিদম হবে Naive Bayes এবং সেই অ্যালগরিদমের একটি মডেল হল Gaussian Naive Bayes। আমরা একটা ফাঁকা মডেল এর অবজেক্ট তৈরি করে নিলাম, তারপর ট্রেইনিং ভ্যালু দিয়ে `fit()` ফাংশন কল করার মাধ্যমে ট্রেইন করলাম।

পরবর্তী চ্যাপ্টারে আমরা দেখব আমাদের তৈরি করা মডেল কেমন পারফর্মেন্স শো করছে!

মডেল পারফর্মেন্স বা অ্যাকুরেসি টেস্টিং - পর্ব ১

আমরা ডেটা সংগ্রহ থেকে মডেল ট্রেইনিং পর্যন্ত কাজ শেষ। এখন শুধু বাকি রইল মডেল কিরকম পার্ফর্মেন্স শো করছে। মডেল পারফর্মেন্স চ্যাপ্টারটা একটু বড় হবে তাই আমি একে দুইভাগে ভাগ করলাম।

দুই পর্বের মডেল পারফর্মেন্স টেস্টিং চ্যাপ্টারের ওভারভিউ

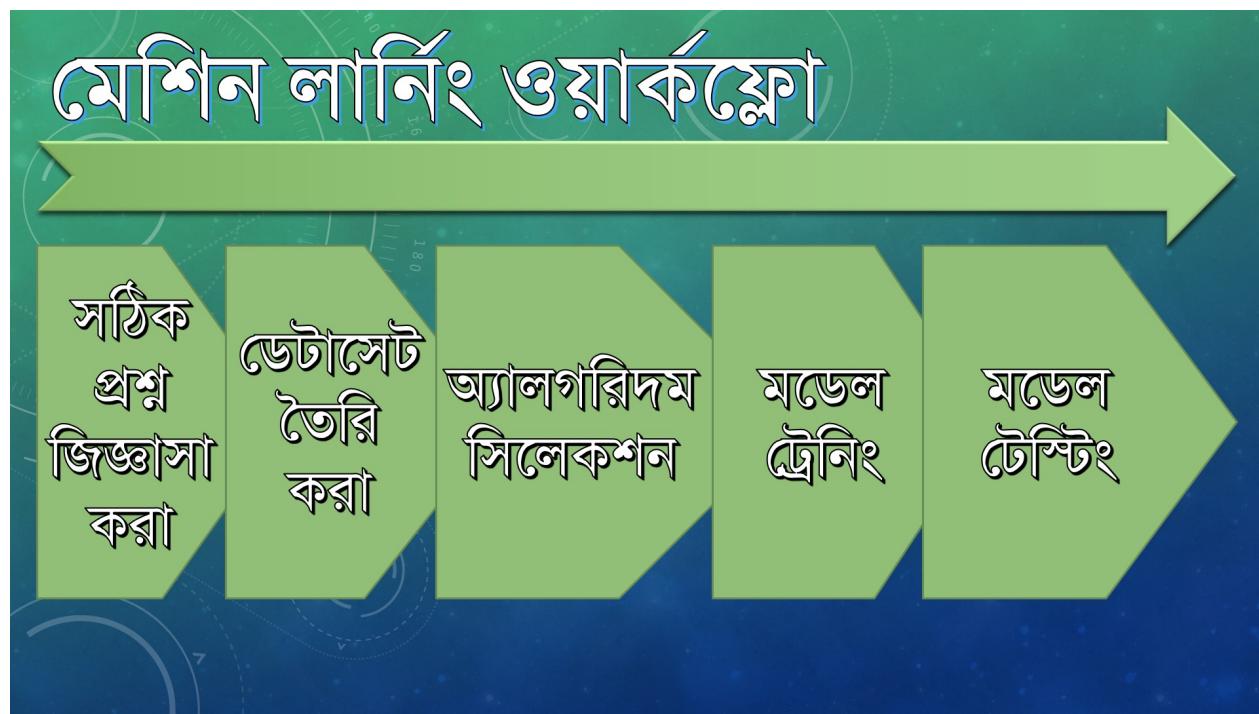
মূল আলোচ্য বিষয়

- টেস্ট ডেটার মধ্যমে মডেল এভালুয়েশন
- ৱেজাল্ট ইন্টারপ্রিটেশন
- ৱেজাল্ট ইম্পুডমেন্ট / মডেল ইম্পুডমেন্ট / অ্যাকুরেসি ইম্পুডমেন্ট

এছাড়াও

- কনফিউশন ম্যাট্রিক্স
 - Recall
 - Precision
 - AUC
 - ROC
- ওভারফিটিং
- মডেল হাইপারপ্যারামিটার
- ওভারফিটিং কমানো
 - K-Fold Cross Validation বা N-Fold Cross Validation
 - Bias-Variance Trade off
 - ভাল পারফর্মেন্সের জন্য কিছুটা পারফেকশন ছাড় দেওয়া

হাটি হাটি পা পা করতে করতে অবশ্যে আমরা চলে এলাম শেষের ধাপে,



তাহলে শুরু করা যাক।

মনে রাখতে হবে,

স্ট্যাটিস্টিক্স গুধু ডেটা নিয়ে কাজ করে, আমরা ডিফাইন করি কোনটা খারাপ আৰ কোনটা ভাল। এবং এই ভাল-খারাপ সম্পূর্ণ নির্ভর করে আমরা মডেল কীভাবে ব্যবহার কৰব।

অনেক থিওরি হল, এবাৰ একটু ইলেক্ট্ৰনিক্সেন্সন দেখি।

যে ডেটায় ট্ৰেইন্ড হয়েছে সেটাতে কেমন পারফর্ম কৰছে

আমরা কাজ শুৰু আগে ডেটাকে দুইভাগে ভাগ কৰেছিলাম, একটা ট্ৰেইনিং আৱেকটা টেস্টিং। এতবাৰ এই কথা দেখতে দেখতে মুখ্যত হয়ে যাওয়াৰ কথা। যাই হোক, আমরা এখন দেখব, যে ডেটায় ট্ৰেইন্ড হয়েছে, তাকে যদি সেই ডেটাই ফিল্ড কৰানো হয়, তাহলে কিৰকম প্ৰেডিক্ষন কৰছে।

ব্যাপৱটা অনেকটা সেই নামতাৰ উদাহৰণেৰ মত,

এটা যদি ট্ৰেইনিং ডেটা হয়:

$$\begin{aligned} 3 \times 1 &= 3 \\ 3 \times 2 &= 6 \\ \dots \\ 3 \times 10 &= 30 \end{aligned}$$

তাহলে ট্ৰেইন্ড ডেটায় কীৰকম ট্ৰেইন্ড হয়েছে সেটা জানাৰ জন্য জিজ্ঞাসা কৰব,

$3 \times 1 = ?$

আপনার মডেলের উপর টাই মারতে নিচের কোডটি বান করুন, (আগে বলা হয় নি যদিও, এই পর্যন্ত Jupyter Notebook এ যত কাজ করেছেন সেখান থেকে কপিনিউট করুন)

```
# This returns array of predicted results
prediction_from_trained_data = nb_model.predict(x_train)
```

এখন `prediction_from_trained_data` ভ্যারিয়েবলে প্রেডিক্ষন করা অ্যারেটা অ্যাসাইন হল। আমরা চাইলে এখন খাতা কলম নিয়ে বসে দেখতে পারি, ডেটাসেট এ প্রতি Observation এ রেজাল্ট কী এবং ডেটাসেট এ প্রতি Observation এ আমাদের তৈরি করা মডেলের প্রেডিক্ষেন রেজাল্ট কী।

অথবা আরেকটা কাজ করা যায়, সাইকিট-লার্ন লাইব্রেরির বিপ্লব ইন মডিউল দিয়ে চেক করতে পারি আমাদের মডেল কয়টা সঠিক ডায়াবেচিস ধরতে পারল আর কয়টা পারল না।

খাতা কলমের বদলে জুপিটার নোটবুক ওপেন করা থাকলে সেখানে লেখা শুরু করুন,

```
# performance metrics library
from sklearn import metrics

# get current accuracy of the model

accuracy = metrics.accuracy_score(y_train, prediction_from_trained_data)

print "Accuracy of our naive bayes model is : {0:.4f}".format(accuracy)
```

যদি ভুলে গিয়ে থাকেন

আমরা ডেটাসেট স্প্লিট করে চারটি ভ্যারিয়েবলে বিশেষজ্ঞান করেছিলাম, `x_train, y_train, x_test, y_test`

যেখানে,

```
x_train = ট্রেইন করার ইনপুট অ্যান্ডগুনো [no_of_preg, insulin, glucose ... etc] (গুরুত্ব ক্ষেত্রে ৯০%)
y_train = x_train এর করেয়াপনিঃ অউটপুট [diabetes -> yes/no] (যেহেতু y_train এর করেয়াপনিঃ অ্যান্ড, বোঝাই যাচ্ছে এটা ৩ ০%)
x_test = টেস্ট করার ইনপুট অ্যান্ডগুনো [গুরুত্ব ক্ষেত্রে ১০% ছিল এবং এই ১০% ট্রেইনিঃ ক্ষেত্রে অস্তিত্ব নাই]
y_test = টেস্ট করার ইনপুটের করেয়াপনিঃ অউটপুট
```

আমরা যেহেতু দেখছি ট্রেইন্ড ডেটায় অ্যাকুরেসি কীরকম, তাই এটা হওয়াই শ্বাভাবিক না যে metrics.accuracy_score ফাংশনে আমরা x_train এ মডেলের আউটপুট (prediction_from_trained_data) এবং x_train এর আসল আউটপুট (y_train)।

আগের কোড স্লিপেটের আউটপুট

আগের কোড স্লিপেটের আউটপুট হল এটা,

```
Accuracy of our naive bayes model is : 0.7542
```

আমাদের সল্যুশন স্টেটমেন্টে টাগেট ছিল ৭০ বা তার বেশি অ্যাকুরেসি তে প্রেভিট করা। কিন্তু এখানে আমরা দেখতে পাচ্ছি অ্যাকুরেসি প্রায় 75%।

থামেন, আগেই সেলিব্রেট করার মত কিছু হ্যান্ড কোড করে আবার সেই ডেটায় প্রেভিকশন টেস্ট করছি। অর্থাৎ সিলেবাসের জিনিসপত্রই জিঞ্জেস করা হল।

টেস্টিং ডেটায় পার্ফর্মেন্স

এবার আপনাকে যদি বলি টেস্টিং ডেটায় পারফর্মেন্স কী হবে সেটার কোড লেখেন, তাহলে আপনি যা করতেন তার সাথে নিচের কোডের মিল আছে কিনা লক্ষ করুন,

```
# this returns array of predicted results from test_data
prediction_from_test_data = nb_model.predict(X_test)

accuracy = metrics.accuracy_score(y_test, prediction_from_test_data)

print "Accuracy of our naive bayes model is: {0:0.4f}".format(accuracy)
```

আউটপুট

```
Accuracy of our naive bayes model is: 0.7359
```

```
# create Gaussian Naive Bayes model object and train it with the data
nb_model = GaussianNB()

nb_model.fit(X_train, y_train.ravel())

Out[37]: GaussianNB()
```

Performance on Training Data

```
In [42]: # this returns array of predicted results
prediction_from_trained_data = nb_model.predict(X_train)

In [52]: from sklearn import metrics

accuracy = metrics.accuracy_score(y_train, prediction_from_trained_data)

print "Accuracy of our naive bayes model is : {0:.4f}".format(accuracy)
Accuracy of our naive bayes model is : 0.7542
```

তার মানে হল সিলেবাসের বাইরে থেকে পশ্চ জিজ্ঞাসা করলেও সে ৭০% অ্যাকুরেসির সাথে উত্তর দিতে পারছে, তারমানে তার দেওয়া উত্তরের ৭৩% সঠিক এবং বাকিটা ভুল।

আমরা এটাই চেয়েছিলাম, অর্থাৎ আমরা যদি এই ট্রেইন্ড মডেলে এবার নতুন পরীক্ষিত কোন ব্যক্তির ডেটা ইনপুট দেই তাহলে তার উত্তর সঠিক হওয়ার সম্ভাবনা ৭০%। যদি মডেল বলে নতুন ব্যক্তির ডায়াবেটিস হতে পারে, তার likelihood হল ৭০%।

কিন্তু

হ্যাঁ ঠিক ধরেছেন, এখনো সেলিব্রেশনের সময় আসে নি। ডেটা কালেকশনের পরবর্তী পেইনফুল কাজ হল পারফর্মেন্স টেস্টিং এবং প্রযোজনীয় পরিবর্তন করা।

রুসিফিকেশন টাইপ প্রবলেমের পারফর্মেন্স টেস্টিং : কনফিউশন ম্যাট্রিক্স

আমাদের সমস্যাটি ক্লাসিফিকেশন টাইপের আর এর জন্য আলাদা কিছু measurement আছে পারফর্মেন্স টেস্ট করার জন্য। যেটার কথা না বললেই নয় সেটা হল Confusion Matrix। নাম শুনে কনফিউজ হওয়ার কিছু নেই। কোড লেখার পশ্চাপাশি আমরা এ বিষয়ে বিস্তারিত জেনে নেব।

আপাতত জেনে রাখুন কনফিউশন ম্যাট্রিক্স দিয়ে আমরা জানতে পারব আমাদের মডেলের পার্ফর্মেন্স কীরকম। তাহলে নিচের কোডটি লিখুন,

```
print "Confusion Matrix"

# labels for set 1=True to upper left and 0 = False to lower right
print "{0}".format(metrics.confusion_matrix(y_test, prediction_from_test_data, labels=[1, 0]))
```

Performance on Testing Data

```
In [63]: # this returns array of predicted results from test_data
prediction_from_test_data = nb_model.predict(X_test)

accuracy = metrics.accuracy_score(y_test, prediction_from_test_data)

print "Accuracy of our naive bayes model is: {0:0.4f}".format(accuracy)
```

Accuracy of our naive bayes model is: 0.7359

```
In [66]: print "Confusion Matrix"

# labels for set 1=True to upper left and 0 = False to lower right
# print "{0}".format(metrics.confusion_matrix(y_test, prediction_from_test_data, labels=[1, 0]))
```

Confusion Matrix

In []:

কনফিউশন ম্যাট্রিক্স

	Predicted True (col 0)	Predicted False (col 1)
Actual True row -> 0	52 (TP)	28 (FP)
Actual False row -> 1	33 (FN)	118 (TN)

আমরা টেবিলের নাম্বারগুলোকে TP, FP, FN ও TN দ্বারা প্রকাশ করতে পারি। যেখানে,

TP = আমন অটেপুট হল > বা ডায়বেটিস হওয়ার প্রভাবনা আছে **এবং** আমাদের টেরি করা মডেলও প্রেডিষ্ট করেছে >
 FP = আমন অটেপুট হল ০ বা ডায়বেটিস হওয়ার প্রভাবনা নাই **কিন্তু** আমাদের টেরি করা মডেল প্রেডিষ্ট করেছে >
 FN = আমন অটেপুট হল > বা ডায়বেটিস হওয়ার প্রভাবনা আছে **কিন্তু** আমাদের টেরি করা মডেল প্রেডিষ্ট করেছে ০
 TN = আমন অটেপুট হল ০ এবং আমাদের মডেলও প্রেডিষ্ট করেছে ০

কনফিউশন ম্যাট্রিক্স একটু কনফিউজিং মনে হলে, ভালভাবে আরেকবার চিন্তা করুন এবং আপনার চিন্তার সাথে সাথে টেবিলটি বুঝতে চেষ্টা করুন।

শর্টকাটে,

- TP = কতগুলা ঘটনা ঘটেছে এবং ঘটেছে হিসেবে ডিটেক্ট করেছে
- FP = কতগুলা ঘটনা ঘটে নাই কিন্তু ঘটেছে হিসেবে ডিটেক্ট করেছে
- FN = কতগুলা ঘটনা ঘটেছে কিন্তু ঘটে নাই হিসেবে ডিটেক্ট করেছে
- TN = কতগুলা ঘটনা ঘটে নাই এবং ডিটেক্ট ও করে নাই

আরেকবার দেখা যাক, তাহলে উপরের স্ট্যাটিস্টিক্স অনুযায়ী,

- 52 টা ঘটনা ডায়াবেচিস হিসেবে ডিটেক্ট করেছে এবং 52 জন আসলেই ডায়াবেচিসে আক্রান্ত <- TP
- 28 টা ঘটনা ডায়াবেচিস হিসেবে ডিটেক্ট করেছে কিন্তু গুঁ 28 জন আসলে ডায়াবেচিসে আক্রান্ত নয় <- FP
- 33 টা ঘটনা ডায়াবেচিস হিসেবে ডিটেক্ট করে নাই কিন্তু গুঁ 33 জন আসলে ডায়াবেচিসে আক্রান্ত <- FN
- 118 টা ঘটনা ডায়াবেচিস হিসেবে ডিটেক্ট করে নাই এবং গুঁ 118 জন ডায়াবেচিসে আক্রান্ত নয় <- TN

যদি আমাদের মডেল ১০০% অ্যাকুরেট হত তাহলে কীরকম হত তার কনফিউশন ম্যাট্রিক্স?

সহজেই বোঝা যাচ্ছে, 100% Accurate Model এর ক্ষেত্রে $FP = 0$ এবং $FN = 0$ হবে। তাহলে কনফিউশন ম্যাট্রিক্স হবে এইরকম,

	Predicted True	Predicted False
Actual True	80 (TP)	0 (FP)
Actual False	0 (FN)	151 (TN)

কনফিউশন ম্যাট্রিক্স পর্যালোচনা : ক্লাসিফিকেশন রিপোর্ট

কনফিউশন ম্যাট্রিক্স এর মাধ্যমে মডেল অ্যাকুরেসি বের করার জন্য আমরা আরও কিছু স্ট্যাটিস্টিক্স রিপোর্ট দেখতে পারি। সূত্র দেখার পশ্চাপাশি সাইকিট এর বিল্ট ইন ফাংশন দিয়ে কীভাবে বের করা যায় আমরা সেটাও দেখব।

ক্লাসিফিকেশন রিপোর্ট জেনারেট হয় আসলে কনফিউশন ম্যাট্রিক্সের ডেটার উপরে। ক্লাসিফিকেশন রিপোর্ট দেখার জন্য নিচের স্টেটমেন্ট রান করুন,

```
print "Classification Report"

# labels for set 1=True to upper left and 0 = False to lower right
print "{0}".format(metrics.classification_report(y_test, prediction_from_test_data, labels=[1, 0]))
```

রিপোর্ট আউটপুট

```

Classification Report
precision    recall   f1-score   support

      1          0.61      0.65      0.63       80
      0          0.81      0.78      0.79      151

avg / total       0.74      0.74      0.74      231

accuracy = metrics.accuracy_score(y_test, prediction_from_test_data)
print "Accuracy of our naive bayes model is: {:.4f}".format(accuracy)
Accuracy of our naive bayes model is: 0.7359

In [67]: print "Confusion Matrix"
# Labels for set 1=True to upper left and 0 = False to lower right
print "{0}".format(metrics.confusion_matrix(y_test, prediction_from_test_data, labels=[1, 0]))
Confusion Matrix
[[ 52  28]
 [ 33 118]]
```

In []:

In []:

এখানে দুইটা টপিক নিয়ে আমরা একটু আলোচনা করব, একটি হল Precision এবং আরেকটি হল Recall।

Precision বের করার সূত্র

$$Precision = \frac{TP}{TP+FP}$$

অর্থাৎ, পার্ফেক্ট প্রিসিপশনের জন্য আমরা জানি, $FP = 0$, সুতৰাং 100% Accurate Model এর

$$Precision = \frac{TP}{TP + 0} = \frac{TP}{TP} = 1$$

এর অর্থ হচ্ছে Precision এর মান যত বড় ততই ভাল। আমাদের টাগেট থাকবে Precision এর মান যতটা বড় করা যায়। ##### 'Recall' বের করার সূত্র

$$Recall = \frac{TP}{TP+FN}$$

একই ভাবে, 100% Accurate Model এর ক্ষেত্রে

$$Recall = \frac{TP}{TP + 0} = \frac{TP}{TP} = 1$$

আবারও, আমাদের লক্ষ থাকবে Recall এর মান যতটা বাড়ানো যায়।

Precision - 0.61 & Recall - 0.65 খারাপ না, বিস্তু এর মান আরও বাড়ানো যেতে পারে। সেই চেষ্টাই আমরা করে যাব।

পারফর্মেন্স ইম্প্রুভ করার উপায় কী কী?

আমরা নিচের পদ্ধতিগুলোর মাধ্যমে মডেলের পার্ফর্মেন্স বের বাড়াতে পারি

- যে অ্যালগরিদমে আছি সেটা অ্যাডজাস্ট বা মডিফাই করা
- আরও ডেটা জোগাড় করা বা ডেটাফ্রেমের ইম্প্রুভ করা
- ট্রেইনিং ইম্প্রুভ করার চেষ্টা করা
- অ্যালগরিদম চেঙ্গ করা

চলুন আমরা বরং অ্যালগরিদম চেঙ্গ করে দেখি : Random Forest

Random Forest দিয়ে কেন দেখব? কারণ,

- এটা Ensemble Algorithm (সহজ কথায় Advanced এবং Complex)
- ডেটার সাবসেটে অনেকগুলো ট্রি থাকতে পারে
- ট্রি এর রেজাল্ট এভারেজ করে যাতে ওভারফিটিং কট্টোলে থাকে এবং পার্ফর্মেন্স ভাল হয়

আমাদের ডেটা নিয়ে কিছুই করা লাগবে না, যেহেতু আমরা প্রিপসেসিংয়ের কাজ আগেই করে রেখেছি। শুধু নতুন মডেল তৈরি করে ডেটা দিয়ে ট্রেইন করব এবং টেস্ট ডেটা দিয়ে পার্ফর্মেন্স টেস্ট করব।

নিচের কোডটি লিখে ফেলুন,

```
from sklearn.ensemble import RandomForestClassifier

# Create a RandomForestClassifier object
rf_model = RandomForestClassifier(random_state=42)

rf_model.fit(X_train, y_train.ravel())
```

এটা লিখে এন্টার মারলে নিচের মত কোন আউটপুট আসলে বুঝবেন ট্রেইনিং খতম, এবার পার্ফর্মেন্স টেস্ট করার পালা

আউটপুট:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                      oob_score=False, random_state=42, verbose=0, warm_start=False)
```

Random Forest Performance Testing : Predict Training Data

আগের মতই কোড,

```
rf_predict_train = rf_model.predict(X_train)

#get accuracy
rf_accuracy = metrics.accuracy_score(y_train, rf_predict_train)

#print accuracy
print "Accuracy: {:.4f}".format(rf_accuracy)
```

আউটপুট:

```
Accuracy: 0.9870
```

অস্থির! তাই না? ডেটাসেট সে ভালই মুখ্য করেছে। এবাব দেখা যাক টেস্টিং ডেটায় পার্ফর্মেন্স কেমন!

Random Forest Performance Testing : Predict Testing Data

```
rf_predict_test = rf_model.predict(X_test)

#get accuracy
rf_accuracytestdata = metrics.accuracy_score(y_test, rf_predict_test)

#print accuracy
print "Accuracy: {:.4f}".format(rf_accuracytestdata)
```

আউটপুট:

```
Accuracy: 0.7100
```

টেস্টিং ডেটার ক্ষেত্রে অ্যাকুরেসি ৭১% আব ট্রেইনিং ডেটায় ৯৮%। অর্থাৎ সিলেবাসের প্রশ্নে উত্তর ভালই দিতে পারছে কিন্তু এর বাইরে থেকে প্রশ্ন করলে উত্তর বেশ খারাপই আসছে। এর মানে সে রিয়েল ওয়ার্ল্ড ডেটায় ভাল প্রেডিক্ট করতে পারছে না, কিন্তু যে ডেটাসেট এর মাধ্যমে যেটা শিখেছে, সেখান থেকে ভাল প্রেডিক্ট করতে পারছে।

এর চেয়ে তো আমাদের Naive Bayes মডেল ভাল কাজ করছিল! আমাদের টেস্টিং ডেটায় ভাল অ্যাকুরেসি দরকার।

এবাব দেখা যাক Random Forest মডেলের ক্লাসিফিকেশন রিপোর্ট কেমন! আমরা উপর থেকে ধার করা কোড কিছুটা পরিবর্তন করেই বসিয়ে দিতে পারি!

```
print "Confusion Matrix for Random Forest"

# labels for set 1=True to upper left and 0 = False to lower right
print "{0}".format(metrics.confusion_matrix(y_test, rf_predict_test, labels=[1, 0]))

print ""

print "Classification Report\n"

# labels for set 1=True to upper left and 0 = False to lower right
print "{0}".format(metrics.classification_report(y_test, rf_predict_test, labels=[1, 0]))
```

আউটপুট:

```
Confusion Matrix for Random Forest
[[ 43  37]
 [ 30 121]]

Classification Report

precision    recall    f1-score   support
      1          0.59      0.54      0.56       80
      0          0.77      0.80      0.78      151
avg / total       0.70      0.71      0.71      231
```

এখানে দেখুন, precision ও recall এর মানও আমাদের আগের Naive Bayes এর চেয়ে খারাপ।

গুরুত্বপূর্ণ: Overfitting

যখন দেখবেন Training Data ও Testing Data এর Accuracy Score এ মোটামুটি ভালই তফাও, তখনই বুঝবেন আপনার মডেলটি মেশিন লার্নিংয়ের সবচেয়ে ক্লাসিক সমস্যার কবলে পড়েছে। সেটা হল Overfitting, আমাদের Random Forest মডেলটি Overfitting এর ভুক্তভোগী। আমরা পরবর্তী পর্বে Overfitting নামক লার্নিংয়ের দুষ্টচক্র থেকে বের হওয়ার বেশ কিছু পদ্ধতি দেখব।

আপডেটেড নোটবুক ডাউনলোড করুন

এই পর্যন্ত যত কাজ করা হয়েছে আপনি নিজে না করে থাকলেও আমার করা নোটবুকটি ডাউনলোড করে রান করে দেখতে পাবেন। ডাউনলোড করন এখান থেকে।

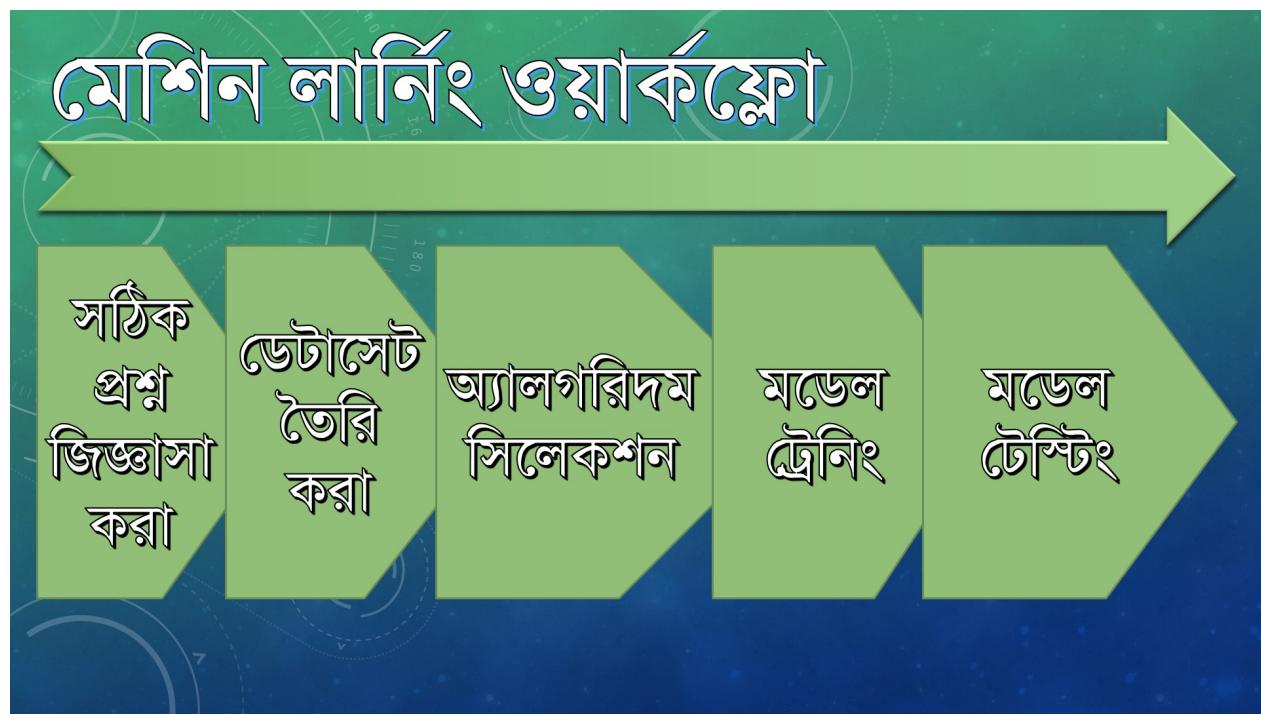
মডেল পারফর্মেন্স - দ্বিতীয় এবং শেষ পর্ব

আমরা মেশিন লার্নিংয়ের একদম শেষ পর্বে চলে এলাম। আগের পর্বেই মোটামুটি দেখেছিলাম মডেল ভাল পারফর্ম না করলে অথবা অন্য মডেল আদৌ ভাল পারফর্ম করছে কিনা সেটা জানার জন্য কীভাবে কাজ আগাতে হয়।

যেসব টপিক আলোচনা করা হয়েছে (টিক দেওয়া) এবং যেসব টপিক নিয়ে আলোচনা করা হয় নাই (টিক ছাড়া)

- টেস্ট ডেটার মাধ্যমে মডেল এভালুয়েশন
- ৱেজাল্ট ইন্টারপ্রিটেশন
- অ্যাকুরেসি বাড়ানো
- কনফিউশন ম্যাট্রিক্স
- Recall
- Precision
- AUC (Area Under Curve)
- ROC (Receiver Operating Characteristics) - Curve
- ওভারফিটিং
- মডেল হাইপারপ্যারামিটার
- ওভারফিটিং মিনিমাইজেশন
- K-Fold / N-Fold Cross-validation
- Bias-Variance Trade Off
- ভাল পারফর্মেন্সের জন্য পার্ফেকশন ছাড় দেওয়া

আমরা এখনো এই ধাপে,



মডেল পারফর্মেন্স রিভিশন - ROC

ROC বোঝার আগে অবশ্যই Confusion Matrix সম্পর্কে জানতে হবে, না জানলে আগের পর্ব থেকে পড়ে নিন।

ROC স্পেসে ROC কার্ড আঁকার জন্য X-axis এ FPR (False Positive Rate) ও Y-axis এ TPR (True Positive Rate) বসাতে হয়।

তারমানে কনফিউশন ম্যাট্রিক্স থেকে প্রাপ্ত TPR ও FPR রেট বসালে আমরা একটা পয়েন্ট পাব, এভাবে একই ডেটাসেটের উপর প্রয়োগকৃত যতগুলো মডেল নিয়ে আমরা কাজ করব সেগুলোর প্রতিটি থেকে একটি করে পয়েন্ট পাব।

এই পয়েন্টগুলো যোগ করে দিয়ে গ্রাফ আঁকলেই আপনি পেয়ে যাবেন আপনার আকঙ্ক্ষিত ROC কার্ড।

একটা পার্ফেক্ট ক্লাসিফিয়ারের TPR হ্য 1 এবং FPR হ্য 0।

একটি উদাহরণ দিয়েই ROC বোঝা যায়।

একটা সিনারিও দেখা যাক,

আমি একটা ডায়বেচিস ডেটাসেট নিলাম, ডেটাসেট এ Observation আছে 1000 টা, আমি এটাকে 80%-20% এ ভাগ করলাম। তারমানে 80% ডেটা হল ট্রেইনিং ডেটা, 20% ডেটা হল টেস্টিং ডেটা।

আবার ধরুন, 200 টা টেস্টিং ডেটার মধ্যে 100 টা হল Positive (মানে আউটপুট পজিটিভ আৱাও সহজভাবে বললে ওঁ ১০০ টা ডেটার আউটকাম হল ডায়বেচিস হয়েছে)। এবং 100 টা Negative।

আমি চারটা মডেল তৈরি করলাম, এই মডেল চারটা আমি ট্রেইন করব ও তাদের পারফর্মেন্স টেস্ট করব। চারটা মডেল হল,

- Gaussian Naive Bayes Model
- Logistic Regression Model
- Random Forest Model
- Artificial Neural Network Model

আমরা এখনো Artificial Neural Network দেখি নাই এবং এটা সম্পর্কে না জানলেও সমস্যা নেই।

আমি আগের পর্বের মত করে প্রতিটা মডেলকে ট্রেইন করে তারপর তাদের Confusion Matrix বের করতে পারি, তাই না? ঠিক সেভাবেই আমি 80% ডেটাসেট দিয়ে মডেলগুলোকে শিখিয়ে পড়িয়ে মানুষ করব তারপর তাদের পারফর্মেন্স টেস্ট করার জন্য পড়া ধরব। (কনফিউশন ম্যাট্রিক্স বের করব)।

আরও মনে করতে থাকেন, প্রতিটি মডেলের Confusion Matrix ও পাশাপাশি তাদের TPR, FPR বের করলাম।

Gaussian Naive Bayes Model

TP = 63	FP = 28
FN = 37	TN = 72

TPR = 0.63
FPR = 0.28

Logistic Regression Model

TP = 77	FP = 77
FN = 23	TN = 23

TPR = 0.77
FPR = 0.77

Random Forest Model

TP = 24	FP = 88
FN = 76	TN = 12

TPR = 0.24
FPR = 0.88

Artificial Neural Network Model

TP = 76	FP = 12
FN = 24	TN = 88

TPR = 0.76
FPR = 0.12

আমরা আগেই জেনেছি ROC Curve এর ক্ষেত্রে Y-axis এ থাকে TPR এবং X-axis এ থাকে FPR। তাহলে আমরা এই চারটা Coordinate সহজেই ROC Space এ বসাতে পারি।

Coordinate গুলো

```
Coordinate -> Model (X, Y)
-----
G point -> Gaussian Naive (.28, .63)
L point -> Logistic Regression (.77, .77)
R point -> Random Forest (.88, .24)
A point -> Artificial Neural Network (.76, .12)
```

এই পয়েন্টগুলো আমরা এখন প্লট করব।

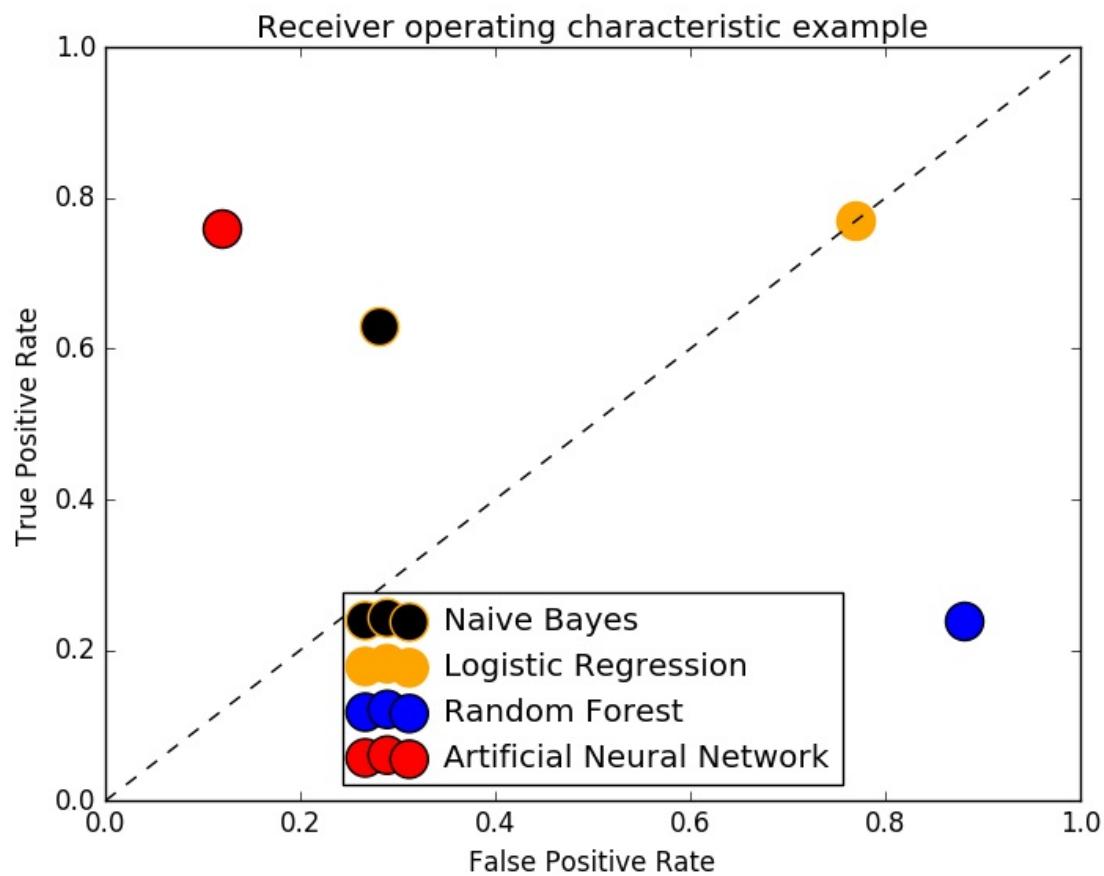
```
import numpy as np
import matplotlib.pyplot as plt

# fpr, tpr
naive_bayes = np.array([0.28, 0.63])
logistic = np.array([0.77, 0.77])
random_forest = np.array([0.88, 0.24])
ann = np.array([0.12, 0.76])

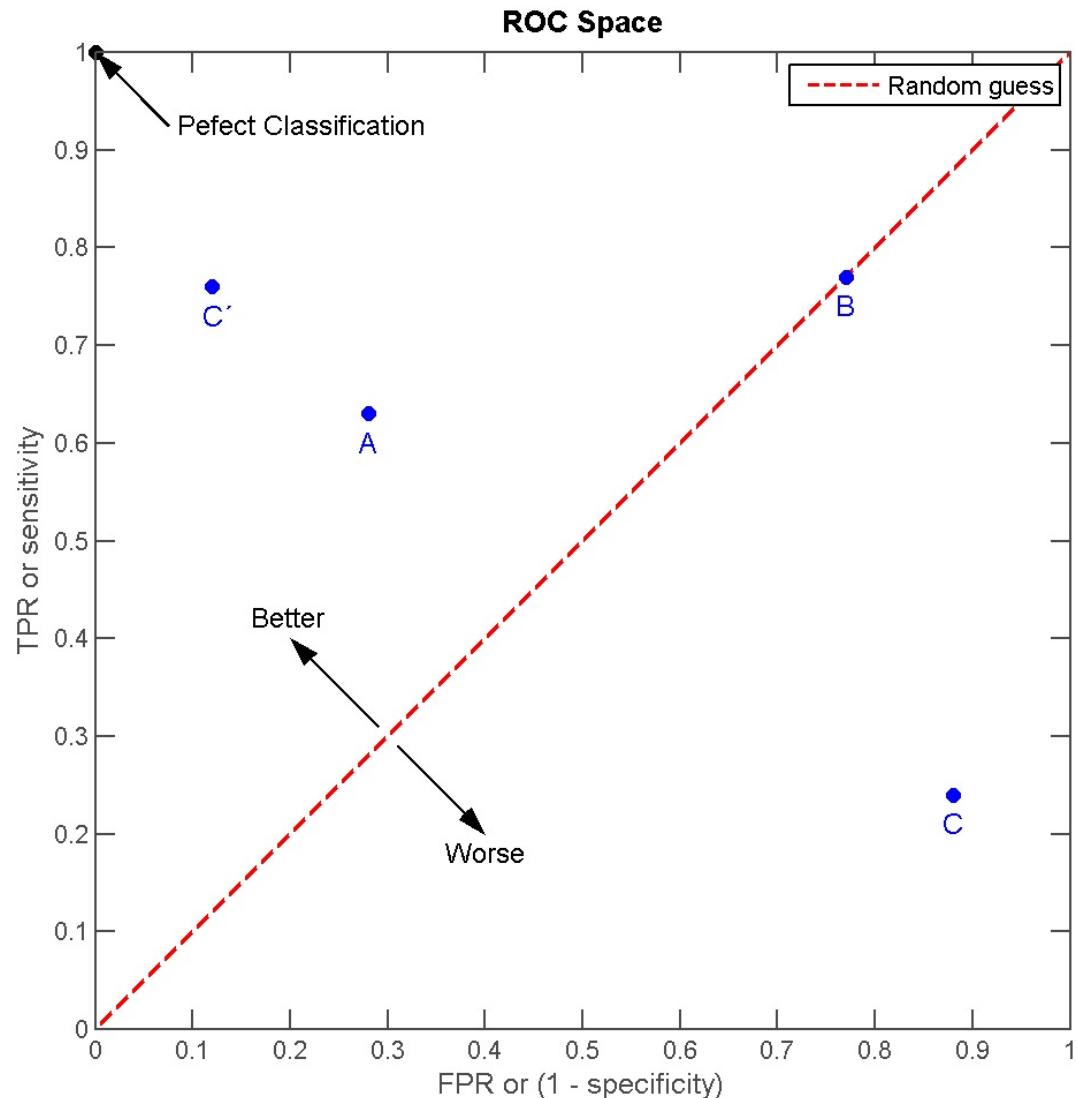
# plotting
plt.scatter(naive_bayes[0], naive_bayes[1], label = 'Naive Bayes', facecolors='black',
            edgecolors='orange', s=300)
plt.scatter(logistic[0], logistic[1], label = 'Logistic Regression', facecolors='orange',
            edgecolors='orange', s=300)
plt.scatter(random_forest[0], random_forest[1], label = 'Random Forest', facecolors='blue',
            edgecolors='black', s=300)
plt.scatter(ann[0], ann[1], label = 'Artificial Neural Network', facecolors='red',
            edgecolors='black', s=300)

plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc='lower center')

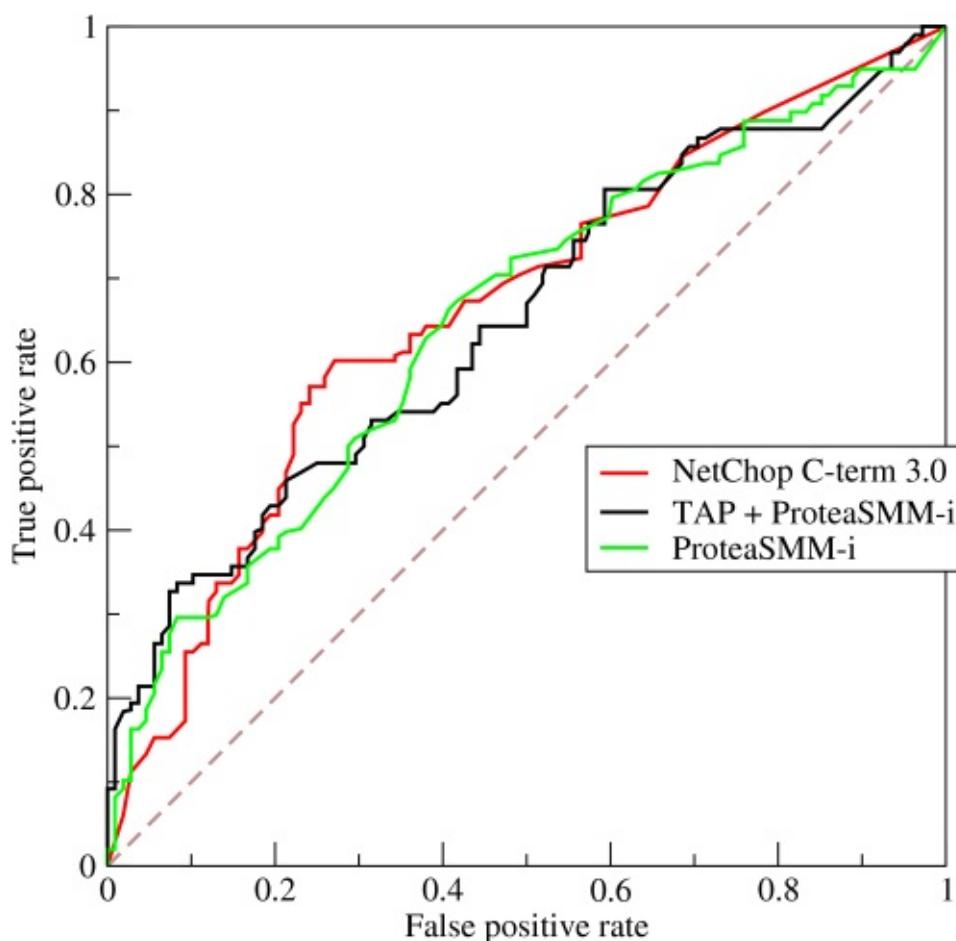
plt.show()
```



উদাহরণটি উইকিপিডিয়া থেকে নেয়া, উইকিপিডিয়ার ROC কার্ডে অতিরিক্ত কিছু জিনিস পয়েন্ট আউট করে দেওয়া আছে,



আমি এখানে প্রতিটি পয়েন্ট বোধানোর জন্য আলাদা ভাবে স্ট্যাটার প্লট করেছি। আপনার যদি মডেল অনেকগুলো হয় কিংবা, একই মডেলের প্যারামিটার পরিবর্তনভিত্তিক পারফর্মেন্স যদি আপনি প্লট করেন তাহলে আপনার প্লট করা ROC কার্ড হবে এইরকম।



আমার এখানে মডেল মাত্র ৪ টা, তাই এখানে লাইন প্লট করলে বোঝা যাবে না তাই, স্ক্যাটার প্লট করা হল।

ROC Curve ব্যাখ্যা

100% Accurate Model এর $FPR = 0$ এবং $TPR = 1$ । এটাকে আইডিয়াল ধরে সহজেই বোঝা যাচ্ছে ANN মডেল হিসেবে সবচেয়ে ভাল, তারপর Naive Bayes, তারপর Logistic Regression এবং সবার শেষে Random Forest পারফর্ম করেছে।

আগেই (এবং আবারো) বলে রাখি, সবসময় ANN > NB > LR > RF এইরকম হবে তা নয়, ডেটাসেট ও প্রবলেমের ধরণ অনুযায়ী এক এক মডেলের পারফর্মেন্স একেক রকম। আমি এখানে পুরো ব্যাপারটা কল্পনা করেছি।

মাঝখান দিয়ে যে ড্যাশড লাইন কোণাকুণি বরাবর গিয়েছে তাকে বলে Line of no-discrimination। পয়েন্ট যত এই লাইনের উপরে থাকবে তত ভাল এবং নিচে থাকলে ততটাই খারাপ।

AUC বা Area Under Curve

উপরে একটা ROC Curve দেখছেন নিচ্য? সেখানে ROC কার্ড যতটা Area কভার করে ততটাই ভাল। 100% Accurate Model এর AUC হল $TPR * FPR$ বা পুরো গ্রাফের ক্ষেত্রফল।

AUC দিয়ে পারফর্মেন্স পরিমাপ করা নিয়ে অনেক পৃশ্ন উঠেছে বর্তমানে, সবাই কমবেশি ROC প্রেফার করে। তাই AUC নিয়ে কথা বাড়ালাম না।

ওভারফিটিং

আগেও বলা হয়েছিল, কোন কোন সময় মডেলের পারফর্মেন্স এতটাই ভাল হয় যে Training Data এর ক্ষেত্রে Accuracy Rate প্রায় 95-99% হয়। কিন্তু Testing Data তে প্রেডিক্ট করতে দিলে 40% Accuracy Rate ও হয় না।

প্রশ্ন হচ্ছে, এটা কেন হয়?

আসলে আমরা যে ডেটাসেট দিয়ে ট্রেইন করি, সেখানে আসল ডেটার পাশাপাশি Noise ও থাকে। অর্থাৎ, 100% Pure Dataset আপনি কখনোই পাবেন না।

একটা ক্লাসিক এক্সাম্পল হতে পারে, আমি কিছু ডেটাসেট জোগাড় করলাম, কয় ঘণ্টা পড়ি আর কয় ঘণ্টা ঘুমাই তার উপর কত মার্কস পাই। এখন আমি এই ডেটাসেট এর উপরে মডেল ট্রেইন করে প্রেডিক্ট করতে বসে যাই এবং যদি কোনভাবে দেখি, পড়া কমিয়ে ঘুমালে মার্কস বেশি আসছে, এবং সেটার উপর ভিত্তি করে আমি পরবর্তী পরীক্ষার আগে ঘুমায়ে কাটালাম কিন্তু পড়লাম না একটুও (কারণ আমার তৈরি A.I. বলেছে ঘুমালে মার্কস বেশি পাওয়া যাবে)। তাতে ফলাফল কী আসবে সেটা বোঝাই যাচ্ছে।

তাহলে এই যে ডুলভাল প্রেডিকশন দিচ্ছে, তার কারণ কী? দুইটা কারণ, (১) পর্যাপ্ত পরিমাণ ডেটা নাই, (২) ডেটাসেট এ কলামের সংখ্যা (ড্যারিয়েবল, এখানে যেমন কয় ঘণ্টা পড়ি আর কয় ঘণ্টা ঘুমাই) কম। মার্কস ভাল আসার অনেক কারণ থাকতে পারে, পরীক্ষা যদি MCQ হয় আর তাতে ঝড়ে বক দিয়ে ভাল পরিমাণ দাগিয়ে ফেললাম, অথবা প্রশ্ন অনেক সহজ হল ইত্যাদি। তাহলে এগুলোতে আমি ইনপুট এ না দিয়েই ট্রেইন করেছি, তাই মডেল স্বত্বাবতার সেই ? কারণ গুলো না জেনেই আমার দেওয়া ডেটাসেট এর সাথে নিজেকে এমন ভাবে খাপ খাওয়াবে তাতে Error সবচেয়ে কম থাকে।

মডেল ট্রেইন মানে হচ্ছে Error কমানো, আর Error কমানোর জন্য প্রতিটি মডেলের হাইপারপ্যারামিটার গুলো ম্যাথমেটিক্যাল অ্যানালাইসিস অনুযায়ী সেট হয়। যে হাইপারপ্যারামিটার ব্যবহার করলে Error সবচেয়ে কম হবে সেটাই মডেল ব্যবহার করবে (এটাই স্বাভাবিক)। কিন্তু Error কম করতে গিয়ে যদি Model, ডেটাসেটের Noise এর সাথে খাপ খাইয়ে নেয় তাহলে যথেষ্ট ঝামেলা হবে।

ওভারফিটিং সম্পর্কে পরবর্তীতে আমরা আরও বিস্তারিত দেখব কয়েকটি ধাপে।

ওভারফিটিং কমানো

ওভারফিটিং কমানোর জন্য যেটা করা যায় সেটা হচ্ছে, ডেটা জোগাড় করা এবং কলামের সংখ্যা বাড়ানো। যতটা পিওর সম্ভব ততটা পিওর ডেটাসেট ও ভাল প্রেডিকশন বেজান্ট দিতে পারে। এটাতো গেল ডেটাসেট এ কি করবেন। চাইলে অ্যালগরিদম টিউন করেও ভাল বেজান্ট বের করা সম্ভব। আমরা একটা মেথড দেখব।

Regularization & Regularization Hyperparameter

একটা অ্যালগরিদম কীভাবে শিখবে সেটা আমরা চাইলে কট্টোল করতে পারি। মেশিন লার্নিং অ্যালগরিদম মানেই তার পিছনে কোন না কোন ম্যাথমেটিক্যাল মডেল কাজ করছে, তাই সেই ম্যাথমেটিক্যাল মডেলের লার্নিং মেকানিজম চাইলে কিছু নির্দিষ্ট প্যারামিটার দিয়ে কট্টোল করা যায়।

ধৰি কোন একটি মডেল আউটপুট বের করে এই সূত্র দিয়ে,

$$Y = ax^3 + bx$$

আমরা এর লার্নিং কট্টোল করার জন্য, ($x * \text{lambda}$) অংশ রেজাল্ট থেকে বিয়োগ দিয়ে Regularized Model তৈরি করতে পারি,

$$Y = ax^3 + bx - \lambda x$$

এখনে, λ ই হল Regularization Hyperparameter।

লক্ষণীয়, Y এর মান আগের প্রেডিকশনের থেকে কিছুটা কম আসবে, তারমানে আমি এবার Training Dataset এই Accuracy আগের চেয়ে কম পাব। কিন্তু এটা ভাল! কারণ? কারণ হচ্ছে, এবার সে প্রতিটা ডেটাসেট মুখ্য করছে না, কারণ Regularization Hyperparameter তাকে মুখ্য করতে দিবে না, x এর মান যত বাঢ়বে, তার প্রেডিক্টেড ভ্যালু ততটাই পেনাল্টি থাবে। একে আমরা তাই Penalized Machine Learning Model বলতে পারি।

যখনই মডেলটা এবার কমানোর জন্য ডেটাসেট এর সাথে খাপ খাওয়াতে যাবে, ওমনি lambda তাকে পেনাল্টি দিয়ে দূরে সরিয়ে দেবে। আমাদের আল্টিমেট কাজ হবে এই lambda কে এমন ভাবে টিউন করা যাতে Testing Dataset এ অ্যাকুরেসি ভাল আসে। Training Dataset এ অ্যাকুরেসি গোলায় যাক :P

Logistic Regression মডেলে Regularization Hyperparameter টিউনিংয়ের মাধ্যমে অ্যাকুরেসি বাঢ়ানো

টপিকের টাইটেল একটু বড় হয়ে গেল। একটু আগে আমরা জানলাম, ম্যাথমেটিক্যাল মডেল হ্যাক করে আমরা Regularization এর মাধ্যমে মডেলের ওভারফিটিং কমাতে পারি। মডেল ভিত্তিক Regularization Hyperparameter বিভিন্ন হয়। সাইকিট লাইব্রেরিতে অলরেডি Logistic Regression এর মডেলের কোড করে দেওয়া আছে এবং তারা Regularization Hyperparameter চেঞ্জ করার জন্য সুবিধাজনক ইটারফেসও দিয়েছে।

আমাদের কাজ হবে, Regularization Hyperparameter এর মান পরিবর্তন করে প্রেডিকশন স্কোর সংগ্রহ করা। তারপর যে Hyperparameter Value তে প্রেডিকশনের অ্যাকুরেসি সর্বোচ্চ হবে সেটা স্টোর করে রাখা।

যিওরি দেখলাম, এবার প্র্যাটিক্যাল দেখার পালা। এখন আপনাকে অবশ্যই নোটবুক বের করে কোড লিখতে হবে।

The screenshot shows a Jupyter Notebook interface with the title "jupyter performance-Copy1". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Help, and various icons for cell operations. A code cell displays Python code for printing a classification report and a confusion matrix for a Random Forest model. The output shows:

```
# labels for set 1=True to upper left and 0 = False to lower right
print "{0}".format(metrics.classification_report(y_test, rf_predict_test, labels=[1, 0]))
```

Confusion Matrix for Random Forest

[[43 37]
[30 121]]

Classification Report

	precision	recall	f1-score	support
1	0.59	0.54	0.56	80
0	0.77	0.80	0.78	151
avg / total	0.70	0.71	0.71	231

In []: # Loi

In []:

```
from sklearn.linear_model import LogisticRegression

lr_model = LogisticRegression(C=0.7, random_state=42)
lr_model.fit(X_train, y_train.ravel())
lr_predict_test = lr_model.predict(X_test)

# training metrics
print "Accuracy : {0:.4f}".format(metrics.accuracy_score(y_test, lr_predict_test))

print "Confusion Matrix"

print metrics.confusion_matrix(y_test, lr_predict_test, labels=[1, 0])

print ""

print "Classification Report"

print metrics.classification_report(y_test, lr_predict_test, labels=[1, 0])
```

আউটপুট

```
Accuracy : 0.7446
Confusion Matrix
[[ 44  36]
 [ 23 128]]

Classification Report
precision    recall    f1-score   support
          1       0.66      0.55      0.60       80
          0       0.78      0.85      0.81      151
avg / total       0.74      0.74      0.74      231
```

এই কাজগুলো আমরা নাইড বেয়েস মডেলের জন্য করেছিলাম। এখানে **C** হচ্ছে আমাদের সেই Regularization Hyperparameter, শুরুতে ধরে নিলাম 0.7 , আমরা পরে এর বিভিন্ন মানের জন্য অ্যাকুরেসি চেক করব।

C (Regularization Hyperparameter) এর মান নির্ণয়

```

C_start = 0.1
C_end = 5
C_inc = 0.1

C_values, recall_scores = [], []

C_val = C_start

best_recall_score = 0

while (C_val < C_end):
    C_values.append(C_val)

    lr_model_loop = LogisticRegression(C=C_val, random_state=42)

    lr_model_loop.fit(X_train, y_train.ravel())

    lr_predict_loop_test = lr_model_loop.predict(X_test)

    recall_score = metrics.recall_score(y_test, lr_predict_loop_test)

    recall_scores.append(recall_score)

    if (recall_score > best_recall_score):
        best_recall_score = recall_score
        best_lr_predict_test = lr_predict_loop_test

    C_val = C_val + C_inc

best_score_C_val = C_values[recall_scores.index(best_recall_score)]

print "1st max value of {0:.3f} occured at C={1:.3f}".format(best_recall_score, best_score_C_val)

%matplotlib inline
plt.plot(C_values, recall_scores, "-")
plt.xlabel("C value")
plt.ylabel("recall score")

```

যেহেতু Regularization Hyperparameter C, আর আমি বিভিন্ন C এর মানের জন্য recall_scores দ্বিতীয়ে চাঞ্চি (recall_score যত বেশি তত ভাল), তাই C_start = 0.1 নিলাম, C_end = 5 নিলাম, আর লুপে C এর মান 0.1 করে বৃদ্ধি করলাম।

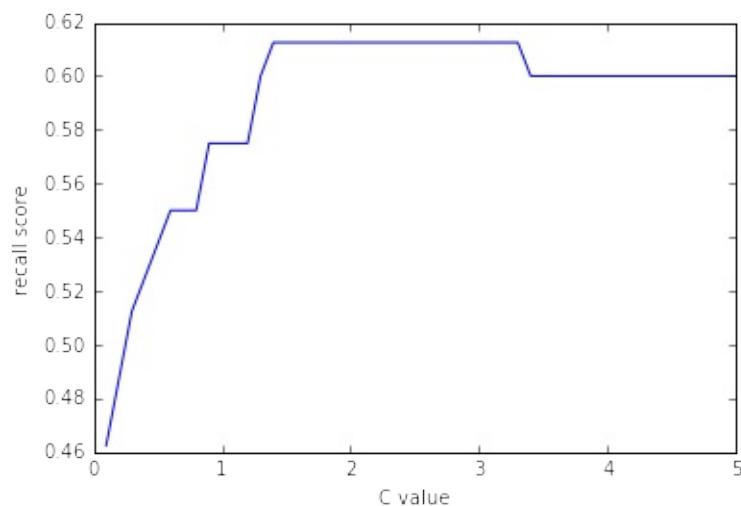
আর প্রতি C এর ড্যালুর জন্য প্রেডিক্ষেড ডেটাসেট দিয়ে অ্যাকুরেসি চেক করলাম, যখনই recall এর মান আগেরটার চেয়ে বেশি হবে তখনই best_recall_score এ recall_score অর্থাৎ বর্তমান স্কোর অ্যাসাইন হবে।

আগের বিষয়গুলো বুঝতে পারলে কোডটা কঠিন কিছু নয়।

C_values এবং recall_scores নামের দুইটা লিস্ট রাখলাম ড্যালু স্টোরের জন্য

আউটপুট

C এর মান বৃদ্ধির সাথে কীভাবে পারফর্মেন্স পরিবর্তন হচ্ছে তার গ্রাফ।



C এর মান যখন 2-3 এর মধ্যে তখন Recall Score সবচেয়ে বেশি, C এর মান 4-5 এবং 0-1 এর মধ্যে কম।

class_weight = 'balanced' ও C পরিবর্তনের সাথে
মডেল পারফর্মেন্স

Regularization Hyperparameter একটাই হবে তার কোন কারণ নেই, একাধিক থাকতে পারে। একটু আগে আমরা C এর মান বের করেছিলাম। এখন আমরা আরেকটি প্যারামিটার (class_weight) কে `balanced` দিয়ে দেখব পারফর্মেন্স কিরকম দিচ্ছে।

`class_weight = 'balanced'` বেথে C এর মান পরিবর্তন করে পারফর্মেন্স বের করাই হবে মূল উদ্দেশ্য।

```

C_start = 0.1
C_end = 5
C_inc = 0.1

C_values, recall_scores = [], []

C_val = C_start
best_recall_score = 0
while (C_val < C_end):
    C_values.append(C_val)

    lr_model_loop = LogisticRegression(C=C_val, class_weight="balanced", random_state=42)

    lr_model_loop.fit(X_train, y_train.ravel())

    lr_predict_loop_test = lr_model_loop.predict(X_test)

    recall_score = metrics.recall_score(y_test, lr_predict_loop_test)

    recall_scores.append(recall_score)

    if (recall_score > best_recall_score):
        best_recall_score = recall_score
        best_lr_predict_test = lr_predict_loop_test

    C_val = C_val + C_inc

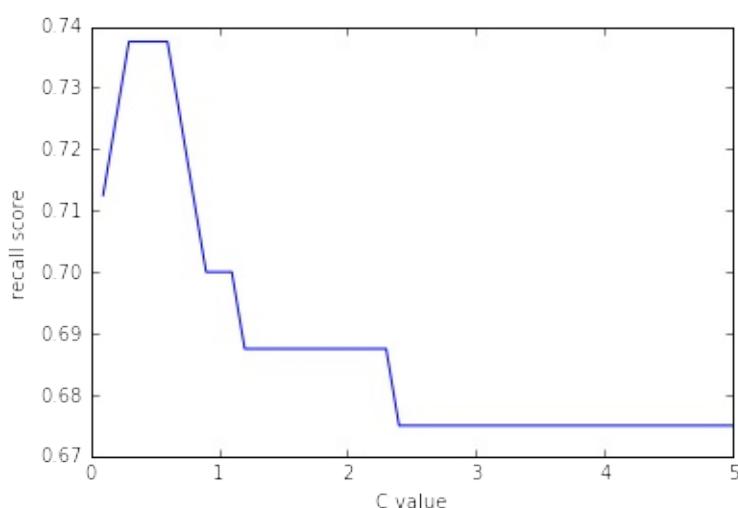
best_score_C_val = C_values[recall_scores.index(best_recall_score)]

print "1st max value of {0:.3f} occurred at C={1:.3f}".format(best_recall_score, best_score_C_val)

%matplotlib inline
plt.plot(C_values, recall_scores, "-")
plt.xlabel("C value")
plt.ylabel("recall score")

```

আউটপুট:



ক্লাস ওয়েট `balanced` দেওয়াতে দেখা যাচ্ছে Recall Score বেড়ে 0.73+ হয়েছে, definitely what we were looking for!

কনফিউশন ম্যাট্রিক্স

কোড:

```
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression( class_weight="balanced", C=best_score_C_val, random_state=42)
lr_model.fit(X_train, y_train.ravel())
lr_predict_test = lr_model.predict(X_test)

# training metrics
print "Accuracy: {:.4f}".format(metrics.accuracy_score(y_test, lr_predict_test))
print metrics.confusion_matrix(y_test, lr_predict_test, labels=[1, 0])
print ""
print "Classification Report"
print metrics.classification_report(y_test, lr_predict_test, labels=[1, 0])
print metrics.recall_score(y_test, lr_predict_test)
```

আউটপুট:

```
Accuracy: 0.7143
```

```
[[ 59  21]
 [ 45 106]]
```

Classification Report

	precision	recall	f1-score	support
1	0.57	0.74	0.64	80
0	0.83	0.70	0.76	151
avg / total	0.74	0.71	0.72	231
	0.7375			

Regularization এর মাধ্যমে এভাবে আমরা অ্যাকুরেসি বাড়াতে পারি (ওভারফিটিং কমিয়ে)।

K-Fold / N-Fold Cross-validation

ওভারফিটিং কমানোর আরেকটি ইফেক্টিভ অ্যালগরিদম হল K-Fold Cross-validation। নামটা অনেক কঠিন শোনালেও কাজ খুবই সহজ।

আমাদের ডায়াবেচিস ডেটাসেট এ কিন্তু মেগেটিভ উত্তর বেশি (মানে ডায়াবেচিস হয় নাই)। যেখানে ডেটাসেট এর ব্যালেন্স কম থাকবে সেসব ক্ষেত্রে K-Fold Cross-validation খুবই ভাল অ্যাকুরেসি দিতে সাহায্য করে।

K-Fold বা N-Fold Cross-validation একই জিনিস যখন $k=N!$ বা $K = \text{Number of observation}$ ।

k-Fold Cross-validation এ যেটা করা হয়, সম্পূর্ণ ডেটাসেটকে k equal sized এ সাবস্যাম্পল করা হয়।

এবাব এই k সংখ্যক সাবস্যাম্পল থেকে একটা একটা করে ডেটা নেয়া হয় টেস্টিং এর জন্য।

যেমন, আমার কাছে 25 টা অবসার্ভেশনের ডেটাসেট আছে, আমি এদেরকে 5 টা গ্রুপে ভাগ করলাম।

তারমানে প্রতিগ্রুপে ডেটাসেট থাকল 5 টা করে। এবাব এই পাঁচটা গ্রুপের প্রথম গ্রুপ আমি Hold করলাম বাকিগুলো ট্রেইনিংয়ে দিলাম, Hold করা ডেটাসেট দিয়ে টেস্ট করলাম।

যিতীয় Pass এ যিতীয় গ্রুপ Hold করব (ট্রেইনিংয়ে পাঠাব না), আৱ বাকিগুলো Training এ পাঠাব।

ঠিক একই ভাবে চতুর্থ এবং পঞ্চম Pass এ ও পজিশনাল গ্রুপটি Hold করে বাকিটা পাঠাব ট্রেইনিংয়ে।

এভাবে 5 বাব 5-Fold এ ট্রেইন করব। যেহেতু প্রতি গ্রুপে Observation 5 টা এবং গ্রুপ সংখ্যা 5 টা তাই এর নাম হবে N-Fold Cross-validation।

Cross-validation ব্যবহাব করে মডেল ট্রেইনিং ও টেস্টিং

ক্রস ভ্যালিডেশন এনাবলড মডেল সাইকিটে বানানোই আছে, যেকোন নৱমাল মডেল এর সাথে CV লাগিয়ে দিলেই Cross-validation Enabled Model পেয়ে যাবেন।

যেমন, LogisticRegression এর Cross-validation Enabled মডেল হবে LogisticRegressionCV, এভাবে বাকিগুলোর জন্যও সত্য।

চলুন এটার পারফর্মেন্স দেখা যাক,

```
from sklearn.linear_model import LogisticRegressionCV

lr_cv_model = LogisticRegressionCV(n_jobs=-1, random_state=42, Cs=3, cv=10, refit=False,
, class_weight="balanced")

# set number of jobs to -1 which uses all cores to parallelize
lr_cv_model.fit(X_train, y_train.ravel())

lr_cv_predict_test = lr_cv_model.predict(X_test)

# training metrics
print "Accuracy: {:.4f}".format(metrics.accuracy_score(y_test, lr_cv_predict_test))
print metrics.confusion_matrix(y_test, lr_cv_predict_test, labels=[1, 0])
print ""
print "Classification Report"
print metrics.classification_report(y_test, lr_cv_predict_test, labels=[1, 0])
```

আউটপুট:

```
Accuracy: 0.7100
[[ 55  25]
 [ 42 109]]

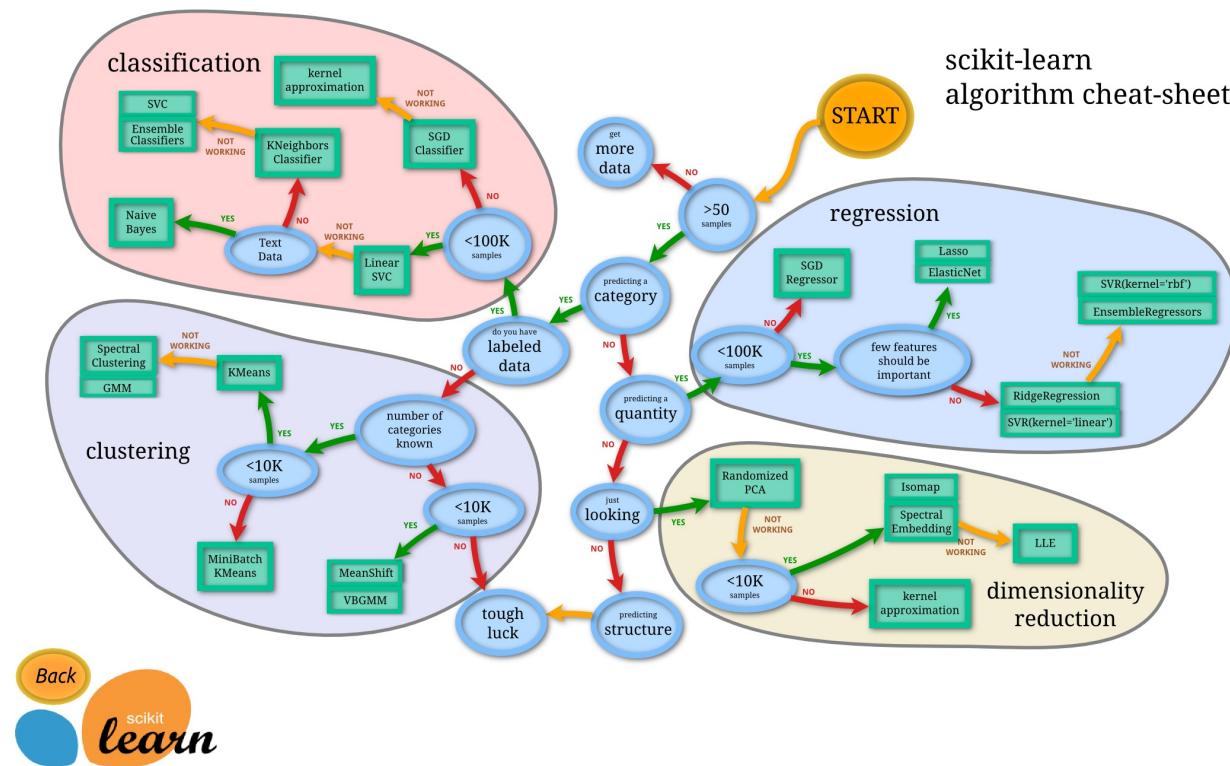
Classification Report
precision    recall    f1-score   support
          1       0.57      0.69      0.62       80
          0       0.81      0.72      0.76      151
avg / total       0.73      0.71      0.72      231
```

10-Fold ক্রস ভ্যালিডেশনে পারফর্মেন্স খারাপ আসে নি!

অনেক বড় হয়ে গেল চ্যাপ্টারটা, তবুও Bias-Variance টা বাদ থেকে গেল। পরবর্তী অন্য কোন পর্বে আমরা দেখব Bias-Variance Trade-off কী জিনিস এবং এর ইম্প্যাছ কতখানি।

Scikit-learn Algorithm Cheat Sheet

ডেটাসেট থেকে অ্যালগো সিলেকশনের উপরে সাইকিটের নিজস্ব একটা চিটশিট আছে। খুবই ইফেক্টিভ,



শেষ প্রত্ন যেসব টপিক নিয়ে আলোচনা করা হল

- টেক্সট ডেটার মাধ্যমে মডেল এভালুয়েশন
- রেজাল্ট ইন্টারপ্রিটেশন
- অ্যাকুরেসি বাড়ানো
- কনফিউশন ম্যাট্রিক্স
- Recall
- Precision
- AUC (Area Under Curve)
- ROC (Receiver Operating Characteristics) - Curve
- ওভারফিটিং
- মডেল হাইপারপ্যারামিটার
- ওভারফিটিং মিনিমাইজেশন
- K-Fold / N-Fold Cross-validation
- Bias-Variance Trade Off
- ভাল পারফর্মেন্সের জন্য পার্ফেকশন ছাড় দেওয়া

লিনিয়ার রিগ্রেশন : প্রাথমিক আলোচনা

এতদিনে আমরা মেশিন লার্নিংয়ের টার্গেট সম্পর্কে জানতে পারলাম। কিন্তু ম্যাথমেটিক্যাল মডেল কীভাবে কাজ করছে সেটা সম্পর্কে এখনো অজ্ঞ। এখনকার আলোচনা গুলোতে প্রেডিচ্টিভ মডেল বিল্ডিংয়ের পাশাপাশি আমরা দেখব মডেলগুলো আসলে কীভাবে তৈরি হচ্ছে বা এর পিছনের লজিক টা আসলে কী।

আজকের আলোচনার বিষয়বস্তু

- লিনিয়ার রিগ্রেশন কী
- মডেল রিপ্রেজেন্টেশন (Model Representation)
- কস্ট ফাংশন (Cost Function)
- কস্ট ফাংশন ইন্টুইশন (Cost Function Intuition)

শুরু করা যাক।

শুরু করার আগে আমরা বিখ্যাত বাড়ির দরদাম ডেটাসেট এর কথা চিন্তা করি।

মনে করুন, আপনার বন্ধু রিয়েল এস্টেট বিজনেসম্যান এবং আপনি একজন ডেটা সায়েন্সিস্ট। আপনার বন্ধু আপনার সম্পর্কে জানতে পেরে ভাবল আপনাকে দিয়ে তার বিজনেসের কিছু কাজ করিয়ে নেবে বলে আপনাকেও কিছু দেবে।

কাজটা হল, আপনার বন্ধুর কাছে একটা ডেটাসেট আছে, যেখানে বাড়ির আকার ও দরদাম দেওয়া আছে। আপনার যেটা করতে হবে, সেটা হল সেই ডেটাসেটে মেশিন লার্নিংয়ের বিশ্লেষণী ক্ষমতা অ্যাপ্লাই করে, যে বাড়ির আকার দেওয়া নেই সেই আকারের বাড়ির দাম প্রেডিক্ট করতে হবে।

আপনার বন্ধুর দেওয়া ডেটাসেট

বাড়ির সাইজ (একক - sq ft) (ধরি এটা X)	বাড়ির দাম (একক - ₹) (ধরি এটা, Y)
2104	399900
1600	329900
2400	369000
1416	232000
3000	539900
1985	299900
1534	314900
1427	198999
1380	212000

1494	242500
1940	239999
2000	347000
1890	329999
4478	699900
1268	259900
2300	449900
1320	299900
1236	199900
2609	499998
3031	599000
1767	252900
1888	255000
1604	242900
1962	259900
3890	573900
1100	249900
1458	464500
2526	469000
2200	475000
2637	299900
1839	349900
1000	169900
2040	314900
3137	579900
1811	285900
1437	249900
1239	229900
2132	345000
4215	549000
2162	287000

1664	368500
2238	329900
2567	314000
1200	299000
852	179900
1852	299900
1203	239500

এই সমস্যাটি আসলে রিগ্রেশন এর মধ্যে পড়ে, কীভাবে?

লিনিয়ার রিগ্রেশন

রিগ্রেশন:

রিগ্রেশন মানে Real-value আউটপুট প্রেডিষ্ট করতে হবে। আরেক ধরণের প্রেডিকশন আমরা করে এসেছি (হাঁ/না ভিত্তিক), সেটা হল ল্যাসিফিকেশন। তারমানে 10, 20, 30, বা 1236, 5.123 ইত্যাদি প্রেডিষ্ট করার মানেই হল আমি একটা রিগ্রেশন প্রবলেমে হাত দিয়েছি।

লিনিয়ার:

লিনিয়ার মানে সরলরেখা টাইপের। আমরা যদি সমস্যাটি একটা লাইনের মত মডেল দিয়ে সল্ভ করতে চাই তাহলে সেটা হবে লিনিয়ার মডেল।

সুতরাং লিনিয়ার রিগ্রেশন

তাহলে লিনিয়ার রিগ্রেশন হল লাইনের মত মডেল দিয়ে Real Value প্রেডিষ্ট করার পদ্ধতি। যদি আমার মডেলটা ব্যাঁকাত্যাড়া লাইনের মাধ্যমে ভ্যালু প্রেডিষ্ট করত তাহলে তার নাম হত Polynomial Regression।

Single Variable Linear Regression

আচ্ছা, ডায়াবেটিস ডেটাসেট বিশ্লেষণ করার সময় আমরা বেশ কিছু ইনপুট ভ্যালু নিয়ে কাজ করেছিলাম যেমন no. of pregnancies, insulin level ইত্যাদি। **কিন্তু আপনার বন্ধু যে ডেটাসেট দিয়েছে তাতে ইনপুট কলাম মাত্র একটা বাড়ি ব প্রাইজ।

তাই আমরা এই সমস্যাকে Single Variable Linear Regression হিসেবে ভাগ করছি। যদি এখানে একটাৱ
বদলে আৱও একটা ইনপুট ড্যারিয়েবল থাকত, যেমন No of rooms থাকত তাহলে তাকে আমরা বলতাম Multi
Variable Linear Regression Problem

এটা একটা Supervised Learning Problem

কাৱণ, আমরা এখানে সঠিক উত্তৰসহ কিছু ডেটা দিচ্ছি, মানে আমরা অলৱেডি কিছু বাড়িৰ আকাৱ ও দাম জানি,
সেটাই মেশিন লাৰ্নিং মডেলে পাঠাব। লেবেলড ডেটা পাঠানোৱ মানেই হল সুপাৱডাইজড লাৰ্নিং।

লিনিয়ার মডেলেৱ মাধ্যমে প্ৰেডিকশন বলতে আসলে কী বুঝাচ্ছি?

আমৰা ছোট একটা ডেটাসেটোৱ এৱ মাধ্যমে বিষয়টা বোঝাৰ চেষ্টা কৰি। ধৰা যাক, আপনাৰ নামীদামী ৰেস্টুৱেটে খেতে
যাওয়াৰ হাৰ আপনাৰ আয়েৰ সমানুপাতিক। আৱ আপনি এমন একটা কোম্পানিতে চাকৰি কৰেন যেখনে আপনাৰ
বেতন মাসে মাসে বাড়ে (আছে নাকি এমন কোম্পানি?)।

আপনি ৫ মাস চাকৰি কৰাৰ পৰ হিসেব কৰতে বসলেন আপনি ৫ মাসেৱ, প্ৰতি মাসে কয়বাৰ কেএফসি, বিএফসি,
হাজীৰ বিৱিয়ানি, স্টাৱ কাৰাৰ ইত্যাদি ইত্যাদিতে খেতে গিয়েছেন। হিসেব কৰাৰ পৰ দেখলেন ডেটাসেট টা দাঁড়িয়েছে
এইৱেকম।

প্ৰতি মাসে আয় (₹)	প্ৰতি মাসে কয়বাৰ বাইৰে খেতে গিয়েছেন
20k	5
30k	10
40k	15
50k	20
60k	25

আপনি যেহেতু এতক্ষণে matplotlib লাইব্ৰেরিতে ভালই হাত পাকিয়েছেন তাই ভাৰলেন একটা গ্ৰাফ এঁকে ফেলা
যাক।

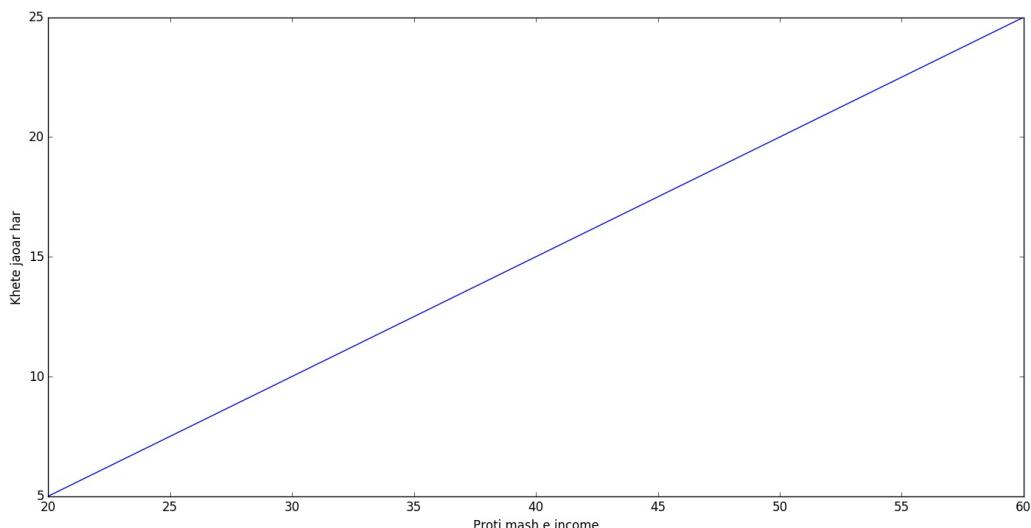
```
# Python 2 & 3
import matplotlib.pyplot as plt
import numpy as np

beton = np.array([20, 30, 40, 50, 60])
khaoa = np.array([5, 10, 15, 20, 25])

plt.xlabel('Proti mash e income')
plt.ylabel('Khete jaoar har')

# আয় vs ক্ষেত্রফল
plt.title("Ae vs Bae")
plt.plot(beton, khaoa)
plt.show()
```

গ্রাফ



এবার যদি আপনাকে বলি, আচ্ছা বলেন তো, ৬ষ্ঠ মাসে আপনি কতবার বাইরে খেতে যাবেন? আপনি কষ্ট ছাড়াই বলে দিতে পারবেন, ৩০ বার (যদি আয় সুষমভাবে বাড়ে)।

এইসব আপনি প্রেডিষ্ট করলেন, সেটা কিন্তু একটা ম্যাথেমেটিক্যাল মডেল তৈরি করা যায়।

$$Khaoa = \frac{Aye - 10k}{10k} \times 5$$

এই সমীকরণ দিয়ে আপনি ডেটাসেট ডেরিফাই করতে পারেন।

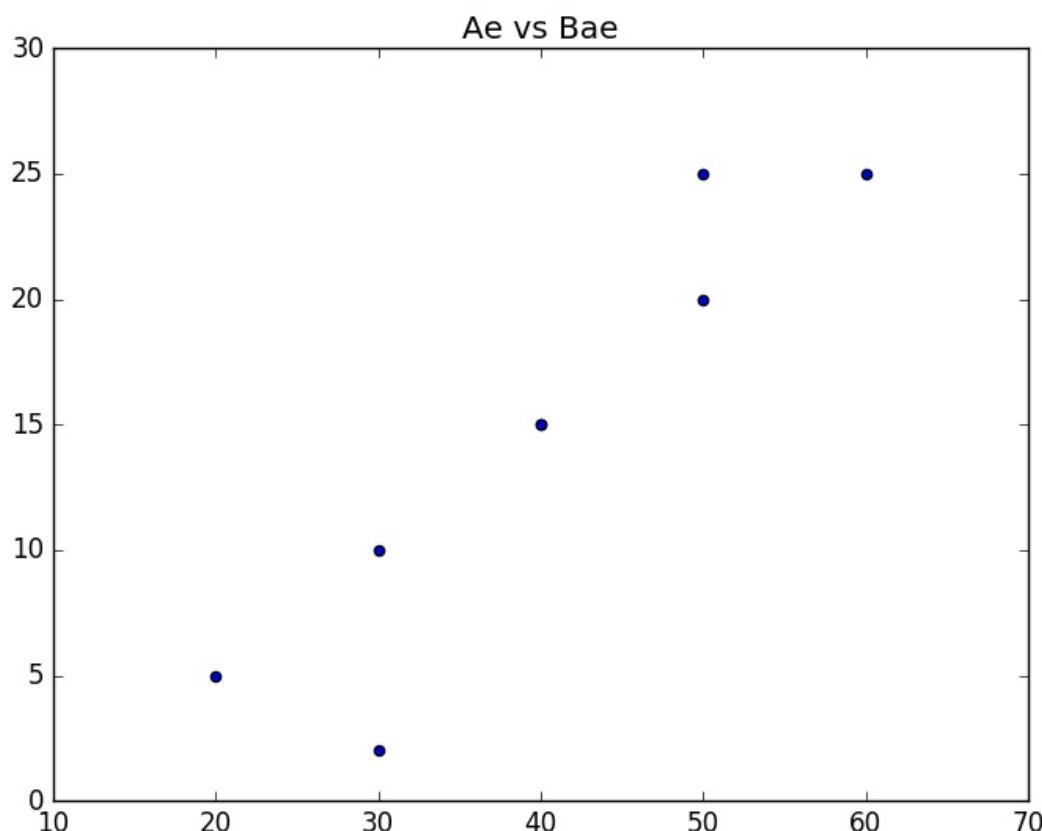
এইখানে আমি একটা সমীকরণ তৈরি করলাম, এটাই হল সেই লিনিয়ার মডেল যেখানে আপনি `Aye` ইনপুট দিলে কয়বার বাইরে খেতে যাবেন সেটা প্রেডিষ্ট করা যাবে। লিনিয়ার মডেল হওয়ার চাক্ষুষ প্রমাণ হল গ্রাফ। আপনার এই ডেটাসেট সন্দেহাতীত ভাবে লিনিয়ার মডেলে `Fit` করে যায়।

এবার আরেকটা সিনারিওর কথা চিন্তা করা যাক,

প্রতি মাসে আয় (₹)	প্রতি মাসে কয়বার বাইরে খেতে গিয়েছেন
20k	5
30k	10
40k	15
50k	20
60k	25
50k	25
40k	15
30k	2

এই ডেটাসেট এ দেখুন প্রথমদিকের মাসগুলোতে আপনার ইনকাম বাড়লেও পরবর্তীতে কমচে, প্রথম প্রথম আপনি বাইরে খেতে যাওয়ার অভ্যাস ছাড়তে পারেন নি। তারপর একদম বেশি কষ্টে নিয়ে ফেলেছেন।

এই ডেটাসেট এর একটা স্ল্যাটার প্লট দেখা যাক,



এইবার আপনাকে যদি বলি, পরের মাসে আপনার আয় যদি 15k তে নামে তাহলে আপনি কয় বার যাবেন? এখন আর ডেটাসেট এ লিনিয়ার প্যাটার্ন নেই, কোন স্পেসিফিক ইকুয়েশন ও নেই যার মাধ্যমে আপনি সহজেই প্রেডিষ্ট করতে পারবেন।

আমরা হ্যত এক্সট্রিম কঙ্কিশন ধরে নন লিনিয়ারিটি বাদ দিয়ে লিনিয়ার মডেল বসাতে পারি। সেটা পরের আলোচনা। এখন আমরা লিনিয়ার প্যাটার্ন নিয়েই আলোচনা করব। আমরা লিনিয়ার রিগ্রেশন বুঝলাম, এখন বুঝব মডেল রিপ্রেজেন্টেশন কী জিনিস।

মডেল রিপ্রেজেন্টেশন

মডেল রিপ্রেজেন্টেশন এর সহজ বাংলা হল, একটা ডেটাসেট এ আমরা যে বিশ্লেষণ চালাব, সেটাৰ বিভিন্ন নোটেশন এর মানে কী, কিভাবে লেখে এবং কেতাবি গঠন কীৱকম। এটাৰ দৰকার কেন? কাৱণ হল আপনি যখন মেশিন লাৰ্নিংয়ের থিওরিটিক্যাল বই পড়তে যাবেন তখন এই কোৰ্সেৰ সাথে মিল পাবেন না। সেখানে ম্যাথেৰ হাবিজাবি সিষ্টল দিয়ে মডেল রিপ্রেজেন্ট কৰা থাকতে পাৰে। তাই সেগুলো বোঝাব জন্য আমাদেৱ অফিশিয়াল নোটেশন সম্পর্কেও জানা দৰকার।

আপনার বস্তুৱ দেওয়া ডেটাসেট টা আবাব একটু দৰকার তাই এখনে আৱেকবাৰ পেষ্ট কৱলাম।

বাড়িৰ সাইজ (একক - sq ft) (ধৰি এটা X)	বাড়িৰ দাম (একক - ₹) (ধৰি এটা, Y)
2104	399900
1600	329900
----	----
1852	299900
1203	239500

এই ডেটাসেট এৰ Row সংখ্যা 47 তাই আমৰা লিখব,

```
m = 47
X = "input" variable / feature
Y = "output" variable / "target" value
```

(x, y)

এই নোটেশন দিয়ে একটা Row বোঝানো হয়, সেটা যেকোন টা হতে পাৰে।

আমি যদি 20 তম Row বুঝাতে চাই সেক্ষেত্ৰে আমি লিখব

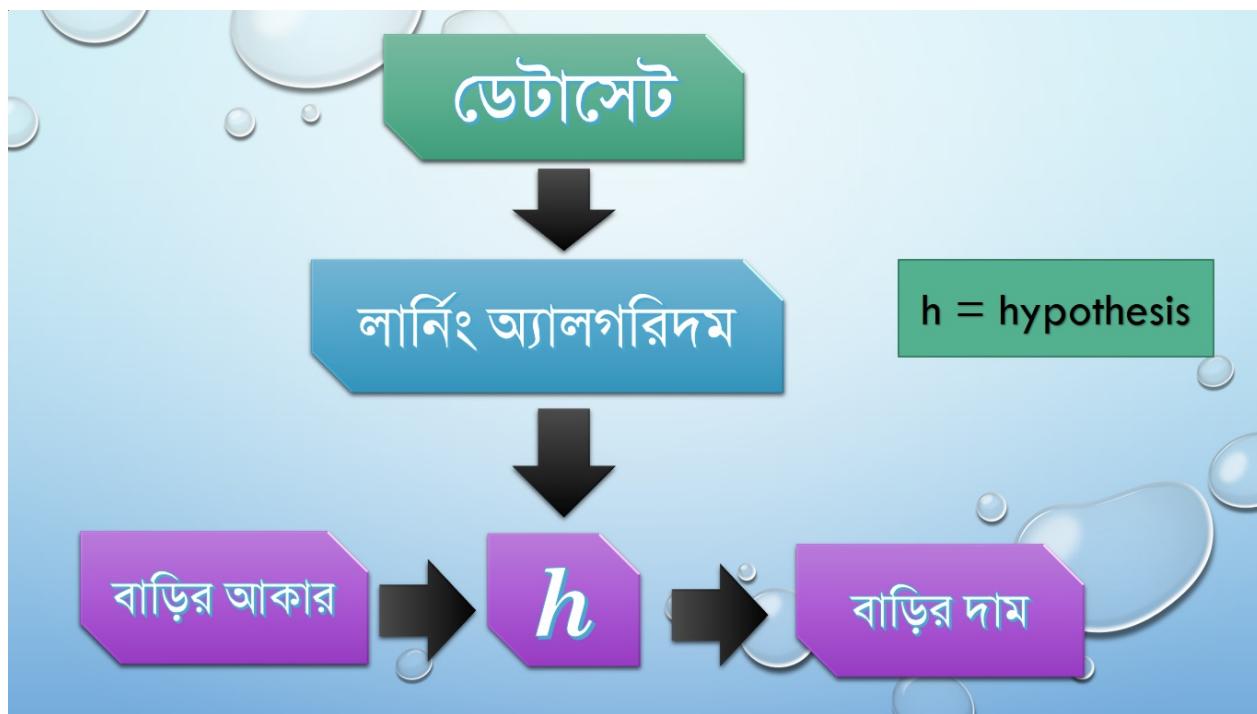
$(X^{(20)}, Y^{(20)})$

|

তাৰমানে i^{th} training example বুঝাতে হলৈ বলতে হবে $(X^{(i)}, Y^{(i)})$

হাইপোথিসিস

একটা ডায়াগ্ৰাম দেখা যাক,



প্রশ্ন হল,

এই h আমরা কীভাবে তৈরি করব? যেহেতু আজকের চ্যাপ্টারটি লিনিয়ার মডেল নিয়ে তাই ধরা যেতে পারে আমরা h একটা লিনিয়ার ফাংশন বাছাই করব।

ধরি আমাদের Hypothesis হচ্ছে, $h_0(x) = \theta_0 + \theta_1 \times x$ শর্টহ্যান্ডে আমরা h_0 কে h লিখে থাকি। লক্ষ্য করে থাকবেন এখানে ইনপুট ড্যারিয়েবল মাত্র একটা, তাই একে আমরা বলব **Univariate Linear Regression**।

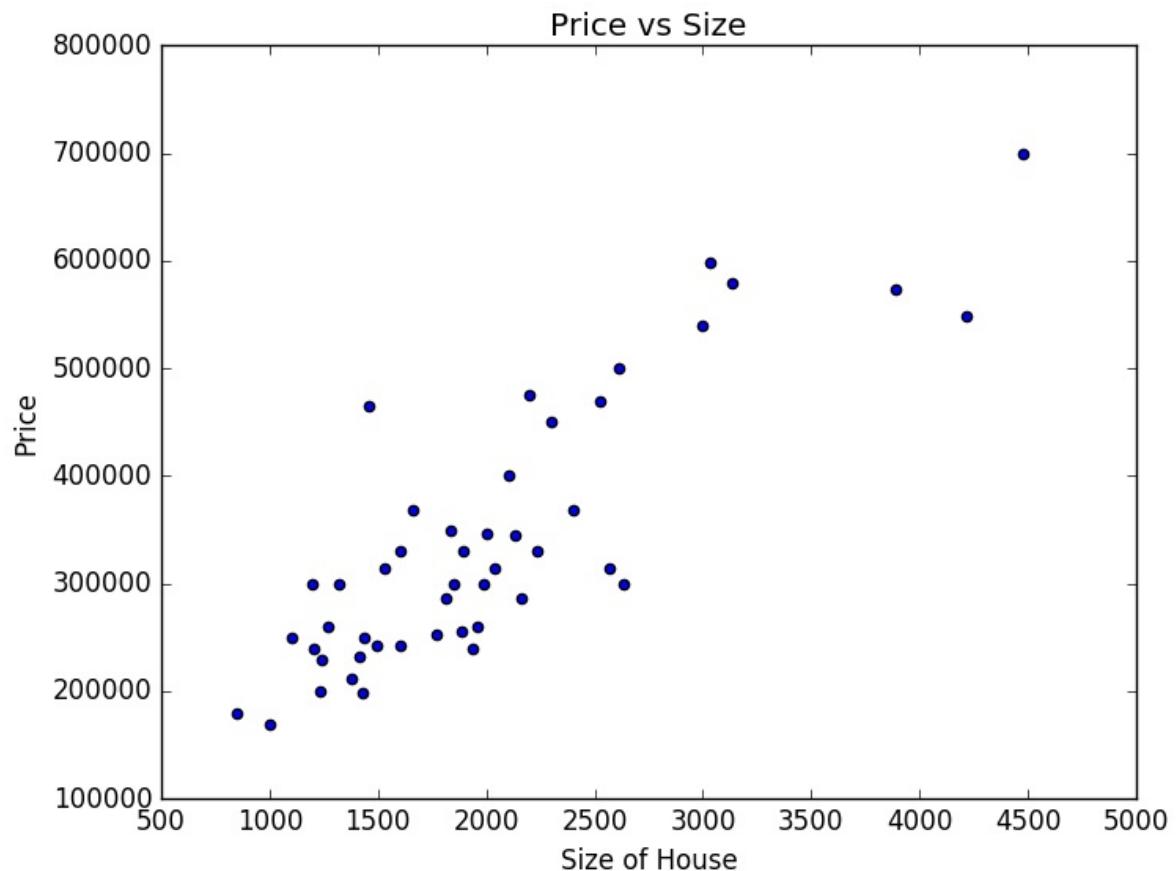
কস্ট ফাংশন (Cost Function)

আমরা সাধারণত আয় হিসেব করে ব্যয় করি। সবসময় চেষ্টা করি যাতে আমাদের ব্যয় সবনিম্ন হয়। মেশিন লার্নিংয়ের ক্ষেত্রে ঠিক তাই করা হয়। এখানে সর্বান্ধক চেষ্টা থাকে, Cost Function কর্তৃত মিনিমাইজ করা যায়। মডেল ট্রেইনিং বলতে আমরা বুঝি Cost Function Minimization।

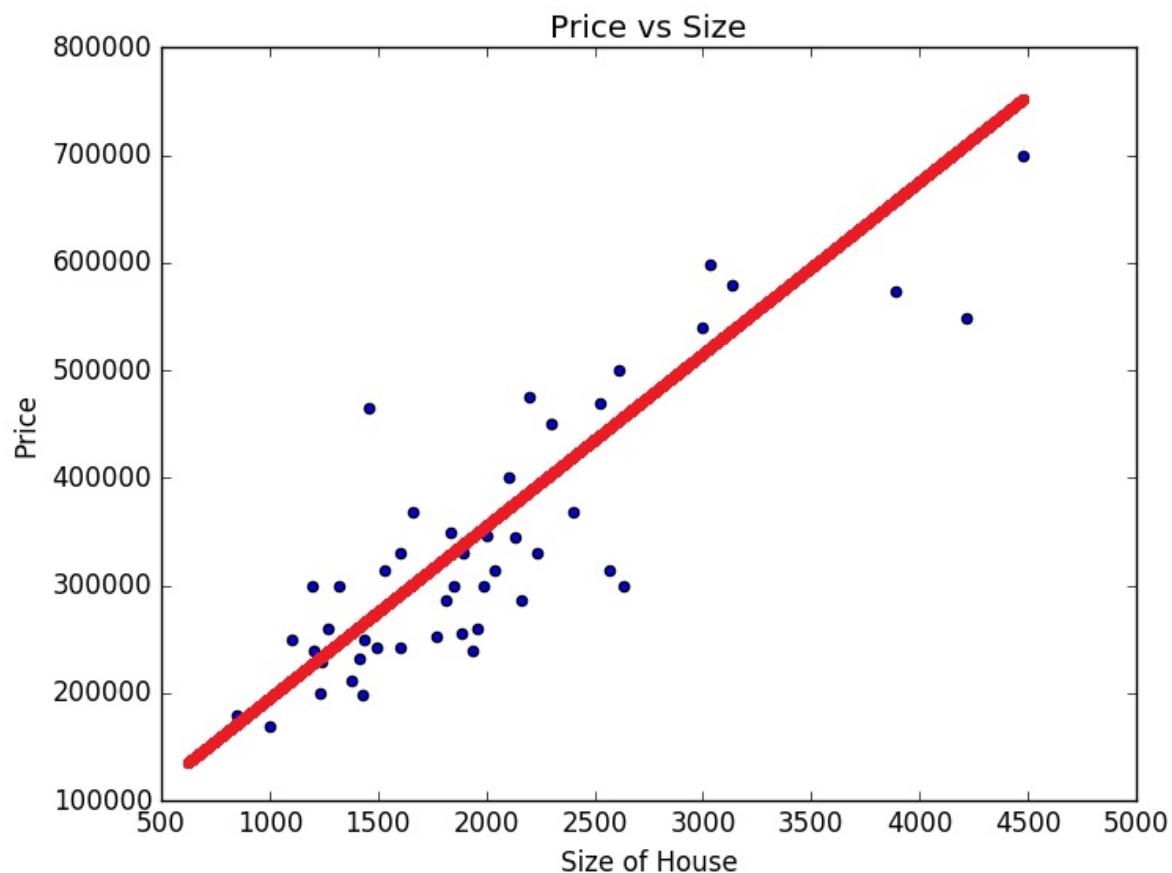
কস্ট ফাংশন মিনিমাইজ করার আগে বুঝতে হবে Cost Function দ্বারা আসলে কী বোঝায়। Cost Function বোঝার আগে আরেকটা জিনিস জেনে নেওয়া যাক।

আমরা হাইপোথিসিস এর জন্য যে ফাংশনটি বাছাই করেছি $h_0(x) = \theta_0 + \theta_1 \times x$ । এখানে শুধু আমরা x এর মান জানি। কিন্তু θ_0 এবং θ_1 এর মান কত হবে সেটা জানিনা। চলুন সেটার মান কী হবে তা নিয়ে একটু গবেষণা করা যাক।

একটা কাজ করি আগে, আপনার বন্ধুর দেওয়া ডেটাসেট এর একটা স্ক্যাটার প্লট এঁকে ফেলি।



আমাদের যেটা করতে হবে এইরকম একটা স্টেইট লাইন এই ডেটার মধ্যে Fit করতে হবে।

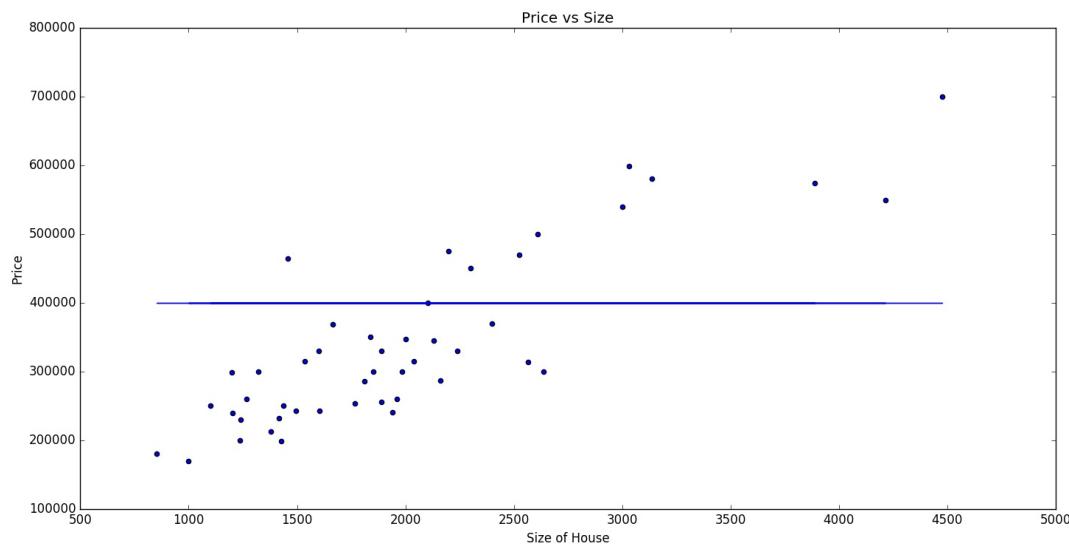


তারমানে θ_0 এবং θ_1 এর মান এমনভাবে বাছাই করতে হবে যেন আমাদের hypothesis এর মান (প্রেডিকশন) ডেটাসেট এর মানের কাছাকাছি হয় অর্থাৎ খরচ সবচেয়ে কম হয় বা Cost Function Minimized হয়।

যিটা এর মান কিরকম হলে সেটা ডেটাসেট এ ফিট করবে সেটা জানার আগে আমাদের দেখতে হবে প্যারামিটার (যিটা গুলো) এর মান পরিবর্তনের সাথে সাথে h এর গ্রাফ কীরকম আসে।

ধরি $\theta_1 = 0$ এবং $\theta_0 = 400000$

তাহলে গ্রাফ আসবে এইরকম,



ধরি $\theta_0 = 0$ এবং $\theta_1 = 120$

তাহলে গ্রাফ আসবে এইরকম,



ডেটাসের এর সাথে কিন্তু প্রায় ফিট করে গেছে। আমাদের যেটা করতে হবে লাইনটা আরেকটু শিফট করে উঠাতে হবে, $y = mx + c$ এর কথা মনে আছে না? m হল ঢাল আর c হল ফ্র্মুলা যার কাজ স্ট্রেট লাইন কে Y-Axis এর পজিচিভ দিকে উঠিয়ে দেয়া (c এর পজিচিভ মানের জন্য)।

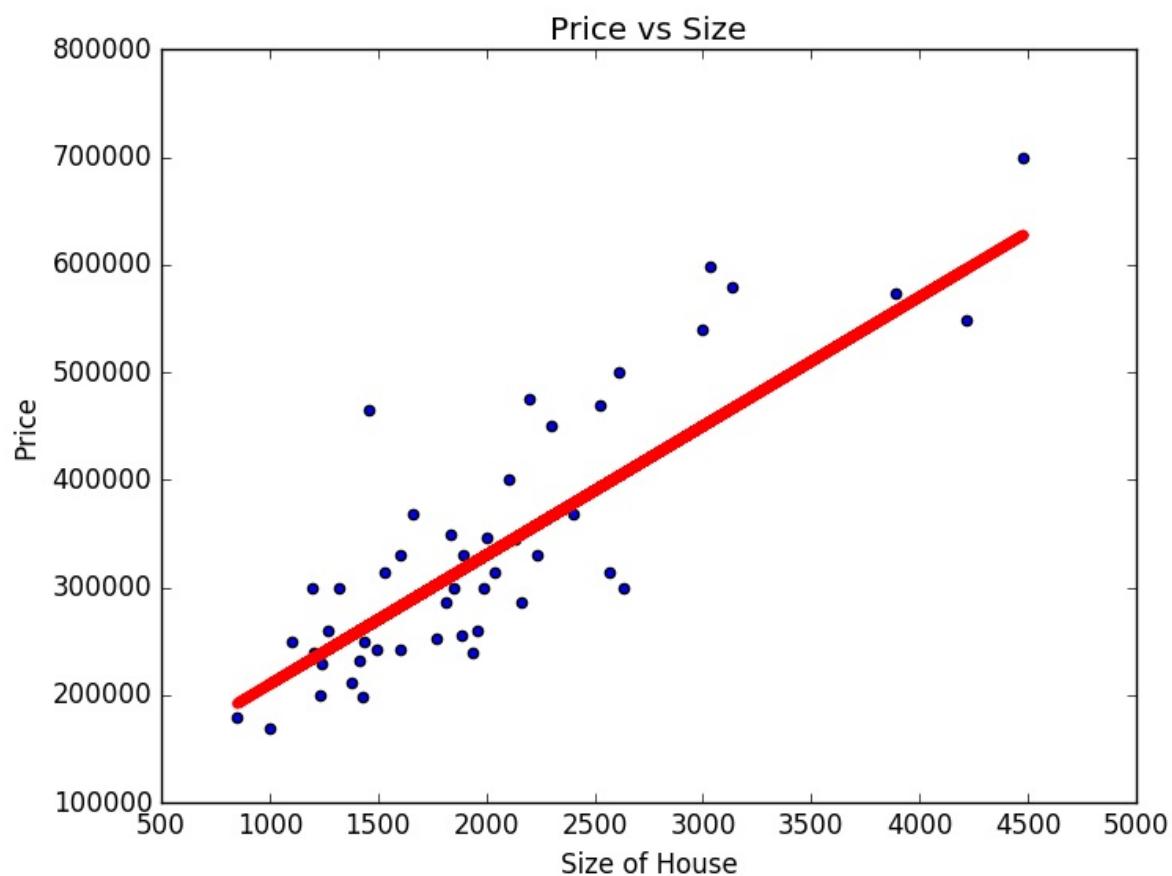
এখনে θ_0 আসলে সেই c এর কাজ করছে এবং θ_1 করছে m এর কাজ।

এবাব আমরা দুইটা প্যারামিটার ব্যবহার করে আবাব গ্রাফ প্রট করাব চেষ্টা কৰি।

ধরি, $\theta_0 = 90000$ এবং $\theta_1 = 120$

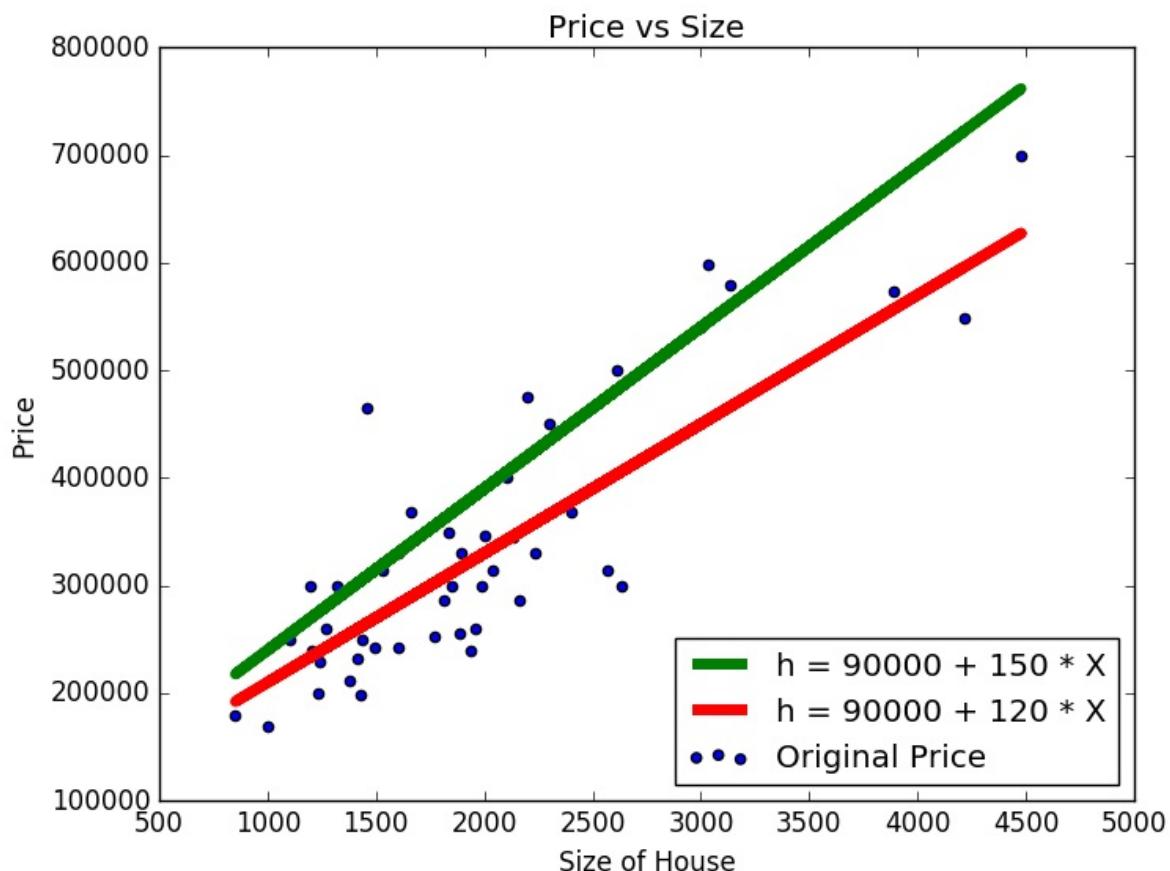
আমাদের হাইপোথিসিস হয়, $h = 90000 + 120 \times x$

এইবাব প্রতি বাড়ির সাইজের জন্য আমাদের হাইপোথিসিস এর আউটপুট এর স্ট্রেইট লাইন প্লট ও স্ক্যাটার প্লটের আউটপুট কম্পেয়ার করে দেখা যাক



বলতেই হবে বেশ ভালই এসেছে (দৃশ্যত)। কিন্তু দেখুন উপরের দিকে কিছু ডেটা বাদ পড়ে গেল, তারমানে আমরা θ_0 ও θ_1 আরেকটু টিউন করে এইরকম গ্রাফ পেতে পারি,

সবুজ রংয়ের লাইনটা আমাদের নতুন হাইপোথিসিস। লাল রংয়ের লাইনটা হচ্ছে আগের।



সব ই বুঝলাম কিন্তু কস্ট ফাংশনটার কথা বললেন না? সেইটার চিকিটাও পাচ্ছি না।

চিত্তার কিছু নাই, আমরা হাইপোথিসিস দাঁড়া করিয়েছি এইবাবে আমরা কস্ট ফাংশন দেখব।

Cost Function কে $J(\theta_0, \theta_1)$ দ্বারা প্রকাশ করা হবে এখানে, যদি আমাদের মডেলে আরও কয়েকটা খিটা থাকত যেমন $\theta_2, \theta_3, \dots$ ইত্যাদি তাহলে আমরা কস্ট ফাংশন প্রকাশ করতাম $J(\theta_0, \theta_1, \theta_2, \theta_3, \dots)$ ইত্যাদি।
তারমানে Cost Function এর প্যারামিটার আর মডেল প্যারামিটার একই হবে।

Cost Function এর সূত্র হল,

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(X^{(i)}) - Y^{(i)})^2$$

মনে আছে তো কোনটা কি? না মনে থাকলে শুরু থেকে পড়ে নিন। আমি এখানে Cost Function হিসেবে Ordinary Least Square মেথড অ্যাপ্লাই করেছি। Cost Function যে শুধু এটাই হবে তা নয়। তবে সাধারণত এইটা ব্যবহার করা হয়।

কস্ট ফাংশন যেটা বলতে চায়, সেটা হল; আমরা প্রতি Observation এর জন্য Error বের করব ($Error = Hypothesis Value - Real Value$), সেই Error কে বর্গ করব। এভাবে প্রতি $Error^2$ এর যোগফল বের করব (m সংখ্যক $Error^2$ এর যোগফল)। তারপর তাকে $\frac{1}{2 \times m}$ দিয়ে ভাগ দিব।

কথা না বাড়িয়ে আমরা ডেটাসেট এর 5 টি অবজারভেশনের জন্য Cost ক্যালকুলেট করি।

ডেটাসেট এর প্রথম পাঁচটি Observation

বাড়ির সাইজ (একক - sq ft) (ধরি এটা X)	বাড়ির দাম (একক - ₹) (ধরি এটা, Y)
2104	399900
1600	329900
2400	369000
1416	232000
3000	539900

ধরি আমাদের **Hypothesis**, $h = 90000 + 120 \times X$

```

Input,
X = [2104, 1600, 2400, 1416, 3000]

Real output,
Y = [399900, 329900, 369000, 232000, 539900]

Hypothesis output,

Example,
h1 = 90000 + 120 * 2104 = 342480
h2 = 90000 + 120 * 1600 = 282000
...
h5 = 90000 + 120 * 3000 = 450000

H = [342480, 282000, 378000, 259920, 450000]

```

তাহলে বোঝা গেল কীভাবে মডেল প্যারামিটারের ভ্যালু ধরে নিয়ে আমরা হাইপোথিসিস ফাংশন দিয়ে ভ্যালু প্রেডিষ্ট করলাম।

ডাল করে দেখুন Real ও Predicted ভ্যালু কাছাকাছি হলেও বেশ এর আছে, আমরা সেই এর এর উপর ভিত্তি করে Cost Calculate করব।

Real Value (Y)	Predicted Value (H)
399900	342480
329900	282000
369000	378000
232000	259920
539900	450000

এখনে Observation বা $m = 5$, কস্ট ফাংশন এর আউটপুট একটু ভেঙ্গে ভেঙ্গে লেখা যাক,

$$J(90000, 120) = \frac{1}{2 \times m} \times (e_1^2 + e_2^2 + e_3^2 + e_4^2 + e_5^2)$$

$$\text{যেখানে, } e_1 = (h(X^1) - Y^1) = 342480 - 399900 = -57420$$

এভাবে আমরা বাকি এর গুলো পাইথন দিয়ে বের করে, সূত্র বসিয়ে Cost Calculate করব।

$$e_1^2 + e_2^2 + e_3^2 + e_4^2 + e_5^2 = 14534002800$$

এবাব $\frac{1}{2 \times m}$ দিয়ে গুণ করে,

$$J(90000, 120) = 1453400280$$

এটা হল $\theta_0 = 90000, \theta_1 = 120$ এর জন্য ক্যালকুলেটেড কস্ট।

এভাবে আমাদের বিডিন θ_0, θ_1 এর কষ্টনেশনের জন্য Cost ক্যালকুলেট করে দেখতে হবে সবনিয় কস্ট কোন কষ্টনেশনে আসে, তারপর সেই কষ্টনেশন দিয়ে আমরা লিনিয়ার মডেল বানিয়ে পার্ফর্মেন্স টেস্ট করব।

সচরাচর জিজ্ঞাস্য প্রশ্ন:

Cost Function এ $\frac{1}{2}$ দিয়ে গুণ করার মানে কী?

এটা করা হয় পরবর্তী ম্যাথমেটিক্যাল ক্যালকুলেশন সহজ করার জন্য। আর কিছুই নয়। আপনি হাফ দিয়ে গুণ না করলেও সমস্যা নাই।

Residual, MSE (Mean Square Error), OLS (Ordinary Least Square), Loss Function, Residual Sum of Squares (RSS) কোনটার মানে কী?

- **Residual** এর মানে সবকয়টি **Observation** এর প্রতিষ্ঠিত ড্যালু আৰ আসল ড্যালুৰ পার্থক্যের যোগফল। **Error** এর সমষ্টি বলতে যা বুঝায়, তা-ই।
- **MSE** বলতে সবকয়টি **Observation** এর **Error** এর বর্গের সমষ্টি বুঝায়
- **Ordinary Least Square** হল **Statistical Estimator**, যার সাহায্যে **Cost** ক্যালকুলেট করছি
- **Loss Function** হল **Cost Function** এর আরেকটা নাম বা **Alias**
- **RSS** এর সূত্র হল $RSS = \sum_{i=1}^m Residual^2$

Cost Function এর মেথড হিসেবে **OLS** ব্যবহার করা হয়েছে কেন?

খুবই গুরুত্বপূর্ণ একটি প্রশ্ন এবং সচরাচর জিজ্ঞাস্য নয়। **Cost Function** হিসেবে OLS ব্যবহার করার মূল কারণ $y = x^2$ এর গ্রাফ **Parabolic** হয়। প্যারাবলিক ফাংশন থেকে **Cost Estimation** করাটা খুবই সহজ। আর $Error^2$ এর গ্রাফ প্যারাবলিক হবে সেটাই স্বাভাবিক।

পরবর্তী পর্বে আমরা কস্ট ফাংশনের আরো ইনট্রুইশন দেখব এবং সমস্যা যদি মাল্টিড্যারিয়েবলের হয় তাহলে লিনিয়ার মডেল দিয়ে কীভাবে রিপ্রেজেন্ট করতে হয় সেটা সম্পর্কেও জানব।

লিনিয়ার রিগ্রেশন : দ্বিতীয় পর্ব

আমরা গত পর্বে লিনিয়ার রিগ্রেশনের বেসিক জানার পাশাপাশি কস্ট ফাংশন ক্যালকুলেশন সম্পর্কে কিছুটা জেনেছিলাম। আজকে আমরা নিচের বিষয়গুলো সম্পর্কে জানার চেষ্টা করব।

আজকের আলোচনার বিষয়বস্তু

- কস্ট ফাংশন ইন্ট্রুইশন - ২ এর গ্রাফ
- গ্রেডিয়েন্ট ডিসেন্ট (Gradient Descent) অপ্টিমাইজেশন

কস্ট ফাংশন ইন্ট্রুইশন

এতক্ষণে লিনিয়ার মডেল সম্পর্কে ভালই ধারণা হয়েছে আশা করি, সেটা যদি হয়ে থাকে আমরা আবেকবার ঘুরে আসি কস্ট ফাংশনের কাছ থেকে।

কস্ট ফাংশনের গ্রাফ দিয়ে লাভ কী?

আমাদের কাজ ছিল কস্ট মিনিমাইজ করা। সকল ইঞ্জিনিয়ারিংয়ের মূল লক্ষ্য তাই। যত কম রিসোর্স ব্যবহার করে যত ভাল ফলাফল পাওয়া যায়। তেমনি মেশিন লার্নিংয়ের জন্য আমাদের মূল লক্ষ্য থাকবে কতটা নির্ভুল প্রেডিকশন করা যায়।

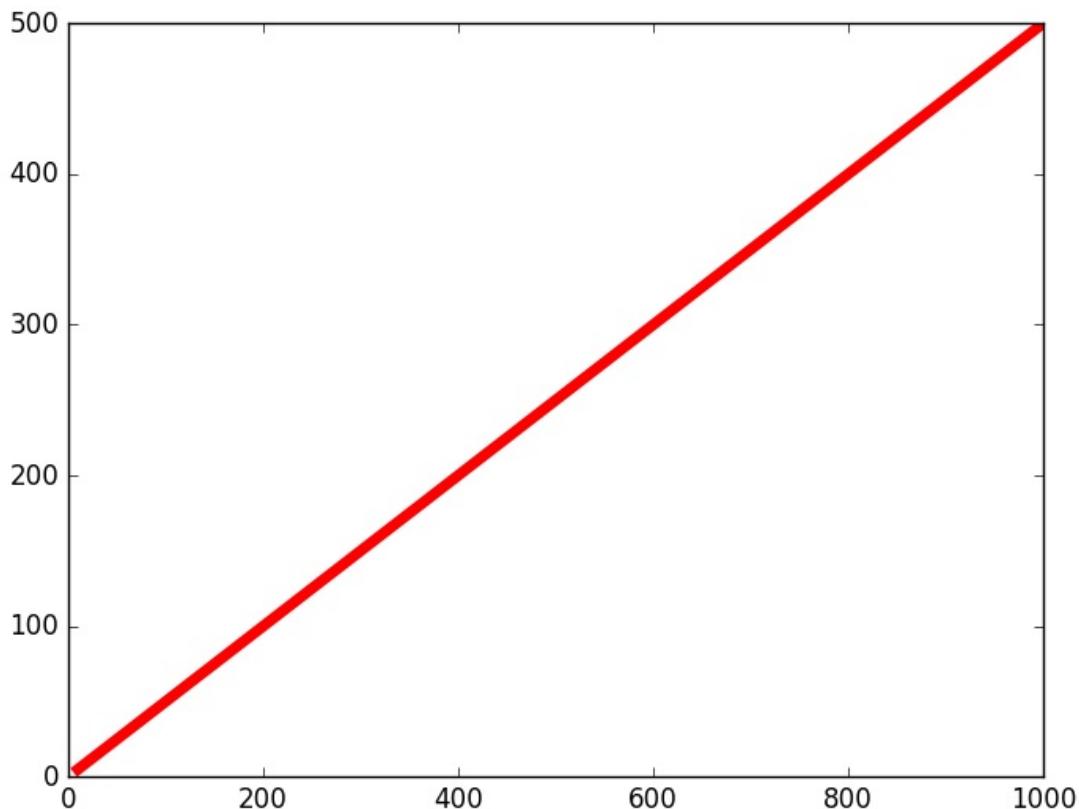
আমরা যদি কতগুলো মডেলের কস্ট ফাংশন এর বেজান্ট স্ট্যাটার প্লট করি তাহলে আমরা গ্রাফ থেকে সহজেই ট্র্যাক করতে পারব সবচেয়ে কম এর কোন প্যারামিটারের জন্য।

সববিছু বাদ দিয়ে নতুন করে একটা জিনিস দেখা যাক, নিচের ডেটাসেট এর কথা চিন্তা করা করি,

আয় (X)	ব্যয় (Y)
10	5
100	50
1000	500

গ্রাফ

এই ডেটাসেটের গ্রাফ এইরকম,



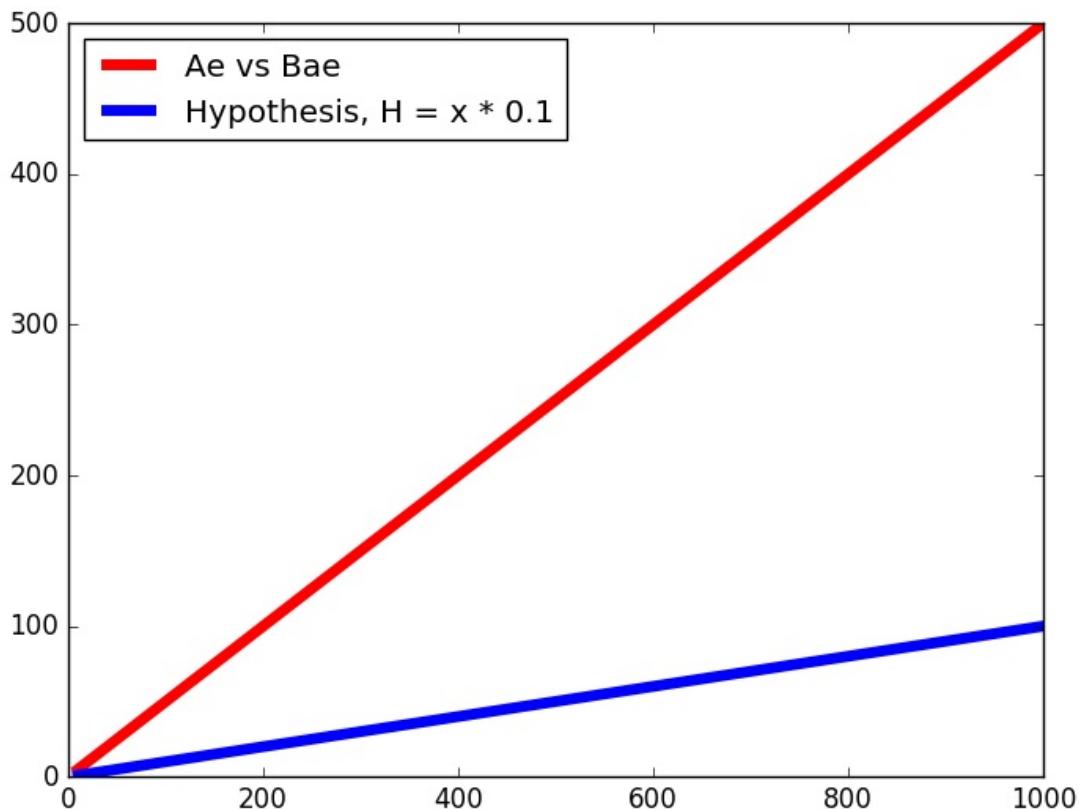
এটা প্রেডিষ্ট করার জন্য আমরা এই মডেল ব্যবহার করব : $h_0(\theta) = \theta \times X$

বিভিন্ন θ এর মানের জন্য আমরা $J(\theta)$ প্লট করব। মানে প্রতি প্রেডিকশনে কষ্ট ক্যালকুলেট করব। তারপর দেখব θ এর কোন মানের জন্য $J(\theta)$ এর মান সর্বনিম্ন আসে।

$$h_0(\theta) = \theta \times X \text{ সাপেক্ষে } J(\theta)$$

ধরি $\theta = 0.1$

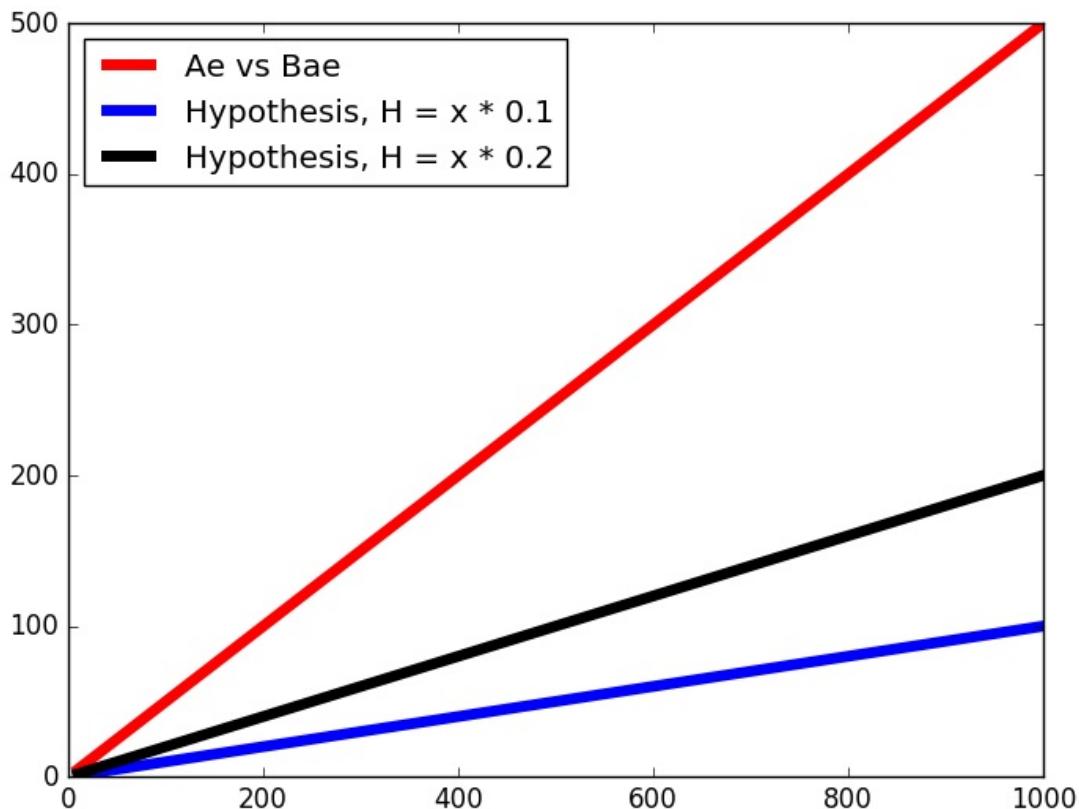
তাহলে প্লট আসবে এরকম,



কষ্ট ক্যালকুলেশন: $J(0.1) = \frac{1}{2 \times 3} \times 4^2 + 40^2 + 400^2 = 26936.0$

আবার ধরি $\theta = 0.2$

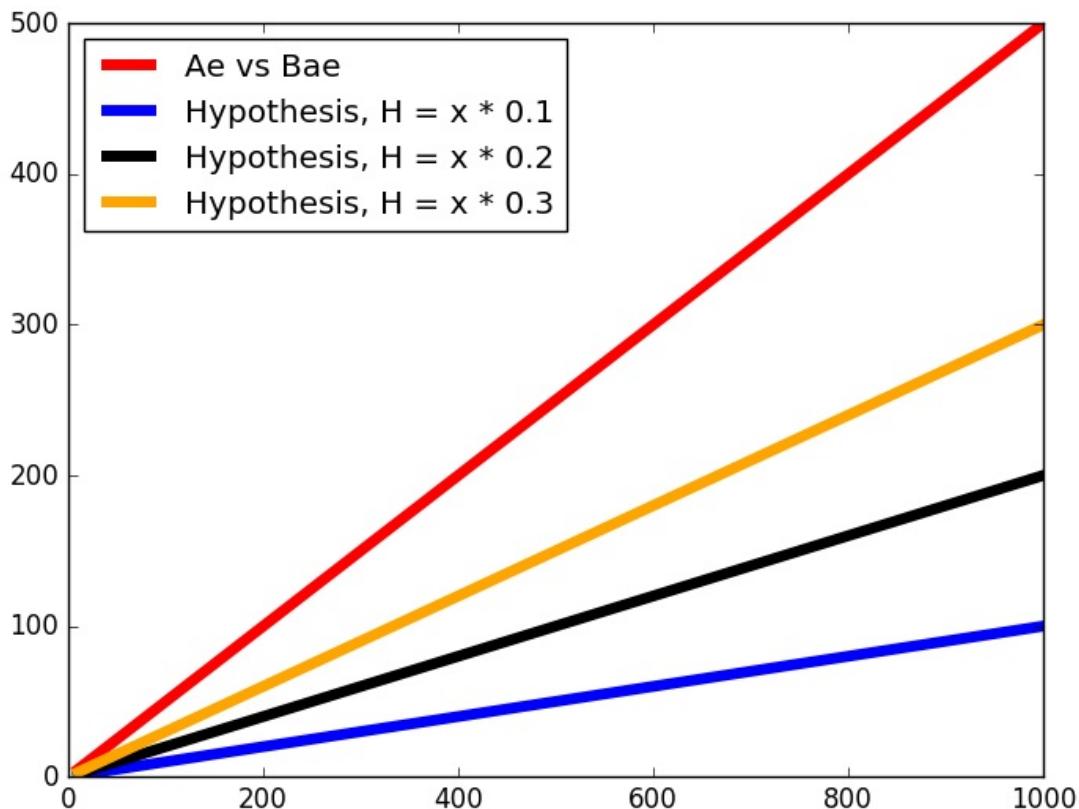
তাহলে প্রট,



কষ্ট ক্যালকুলেশন: $J(0.2) = \frac{1}{2 \times 3} \times 3^2 + 30^2 + 300^2 = 15151.5$

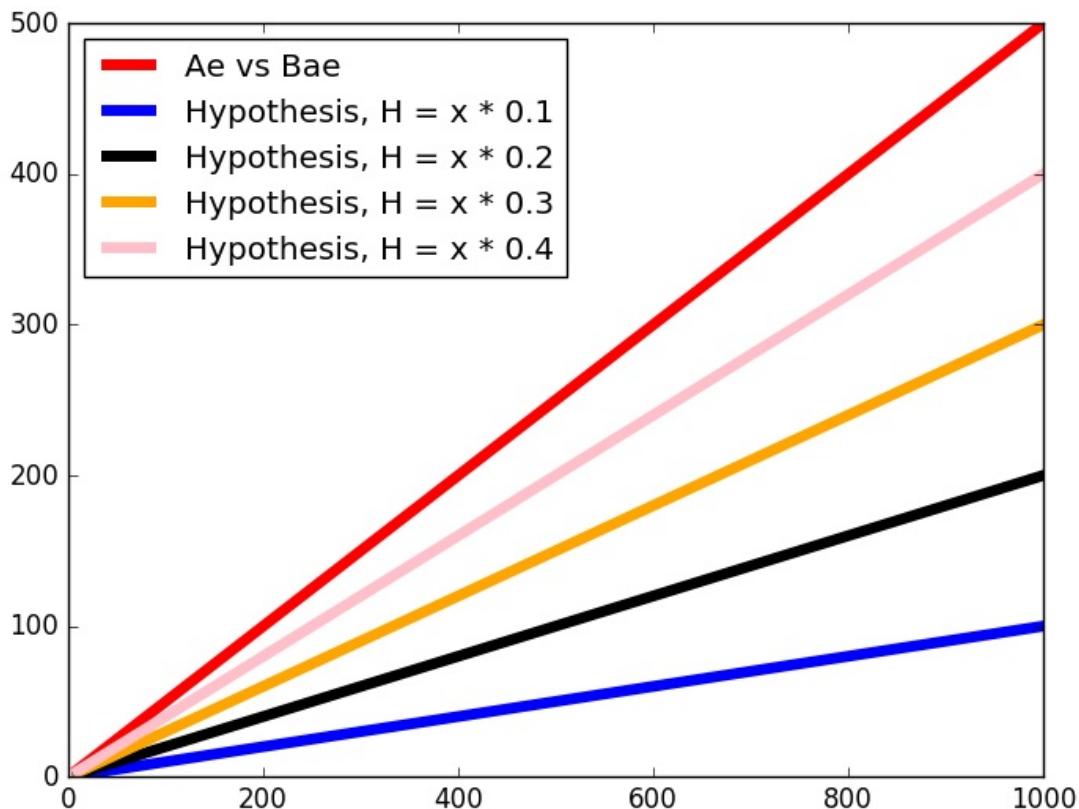
আবার ধরি $\theta = 0.3$

তাহলে প্রট,



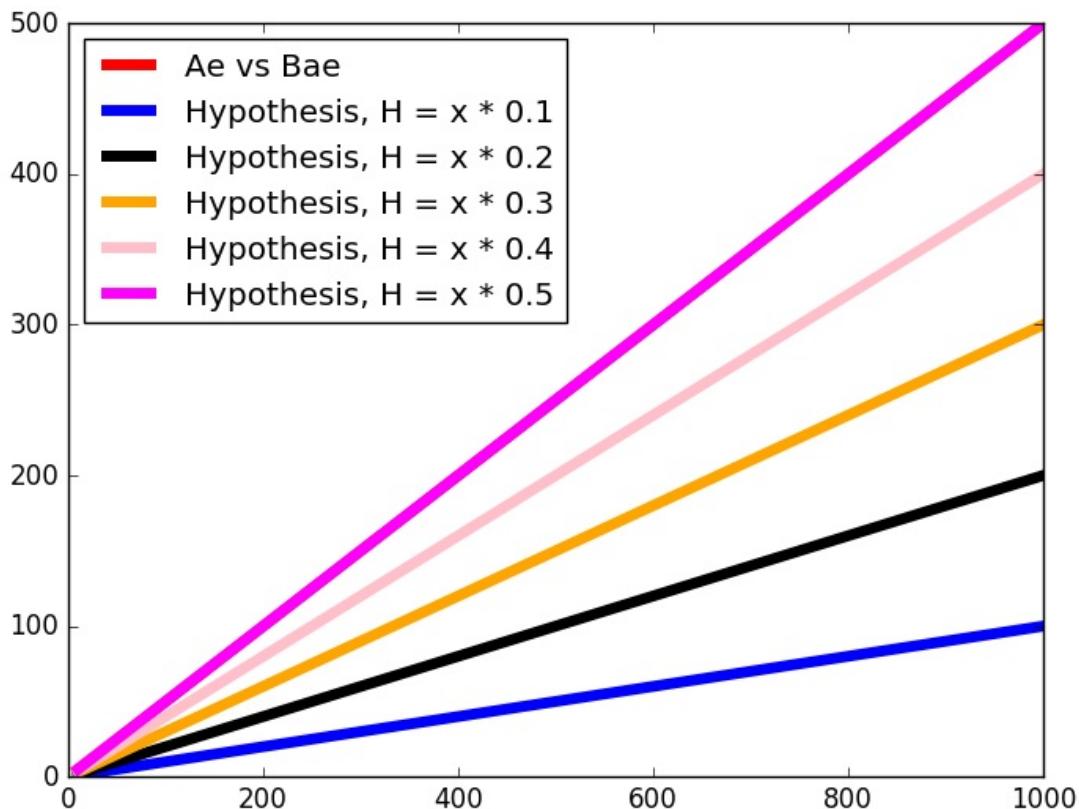
কষ্ট ক্যালকুলেশন: $J(0.3) = \frac{1}{2 \times 3} \times 2^2 + 20^2 + 200^2 = 6734.0$

আবারও ধরি $\theta = 0.4$



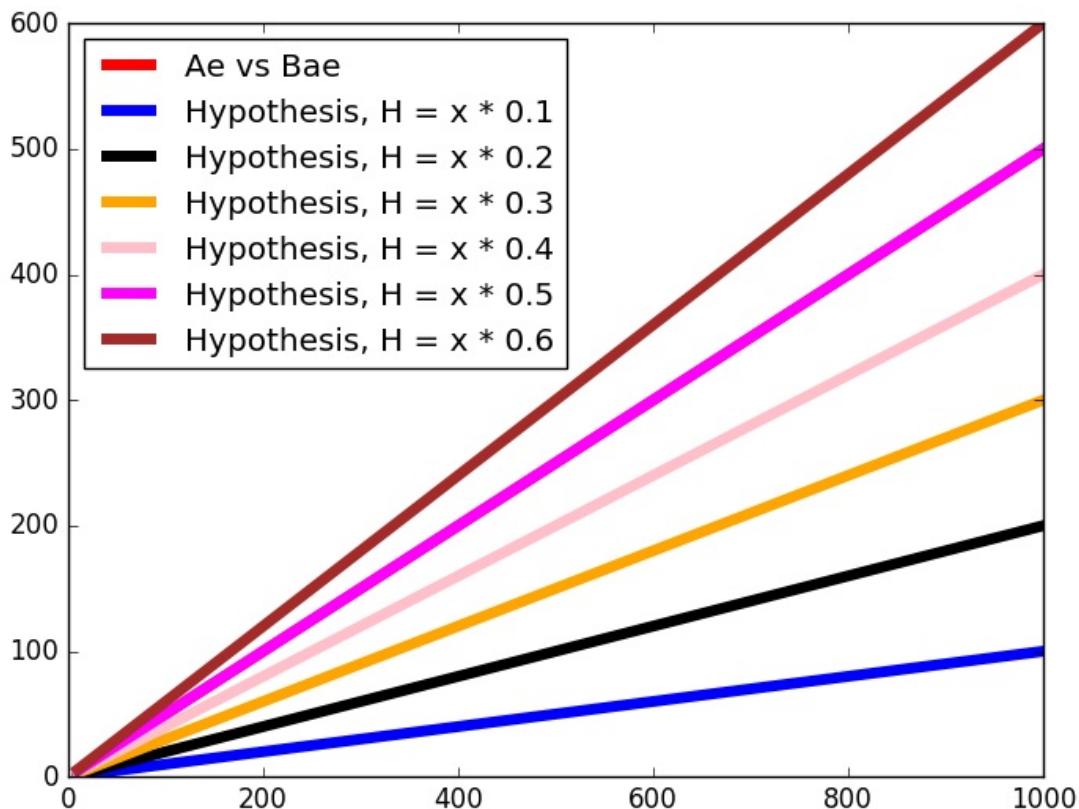
কষ্ট ক্যালকুলেশন: $J(0.4) = \frac{1}{2 \times 3} \times 1^2 + 10^2 + 100^2 = 1683.5$

$$\theta = 0.5$$



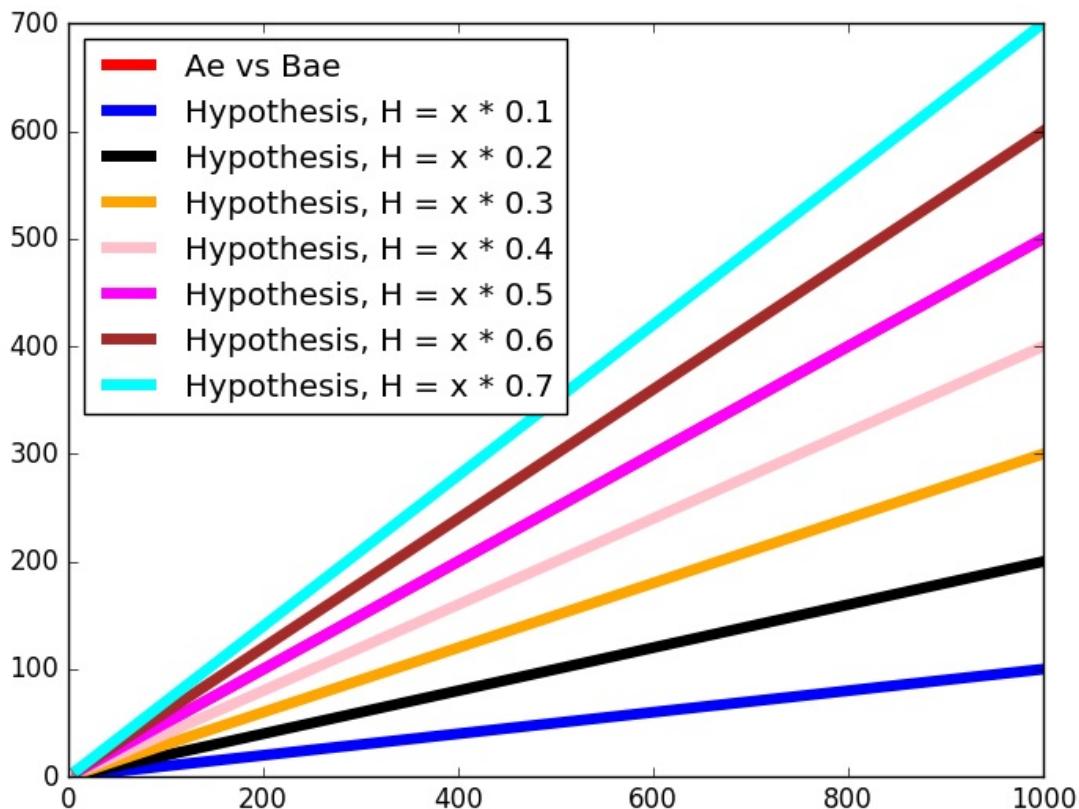
কষ্ট ক্যালকুলেশন: $J(0.5) = \frac{1}{2 \times 3} \times 0^2 + 0^2 + 0^2 = 0$

থিটা এর মান আরও বাড়ালে, $\theta = 0.6$



কষ্ট ক্যালকুলেশন: $J(0.6) = \frac{1}{2 \times 3} \times (-1)^2 + (-10)^2 + (-100)^2 = 1683.5$

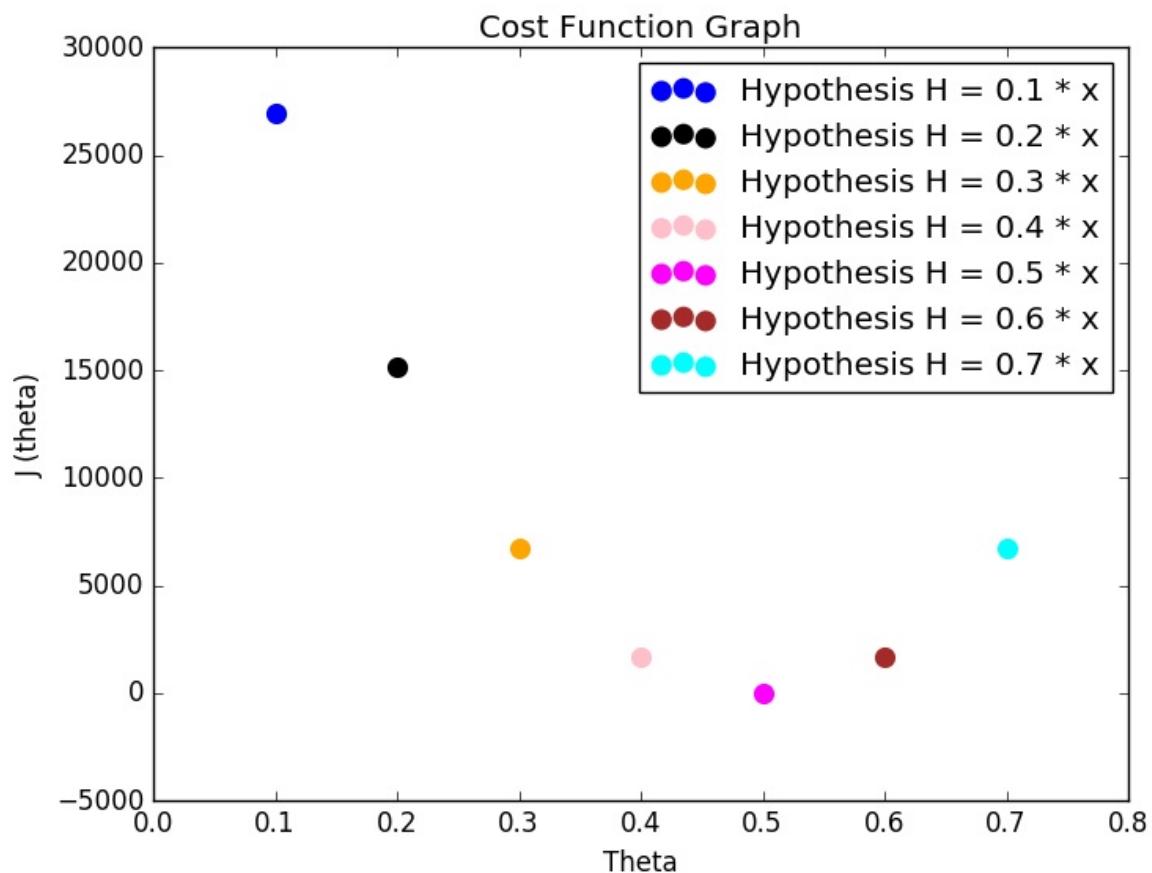
আরও বাড়িয়ে $\theta = 0.7$



কষ্ট ক্যালকুলেশন: $J(0.7) = \frac{1}{2 \times 3} \times (-2)^2 + (-20)^2 + (-200)^2 = 6734.0$

থাক আব বাড়ালাম না, এখন আমরা প্রতি খিটার মানের জন্য যতগুলো $J(\theta)$ এর মান পেয়েছি সেগুলোর স্ট্যাটার প্লট তৈরি করি,

কষ্ট ফাংশন গ্রাফ



Python 2 and 3

```
J = [26936.0, 15151.5, 6734.0, 1683.5, 0, 1683.5, 6734.0]
theta = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7]
colors = ['blue', 'black', 'orange', 'pink', 'magenta', 'brown', 'aqua']

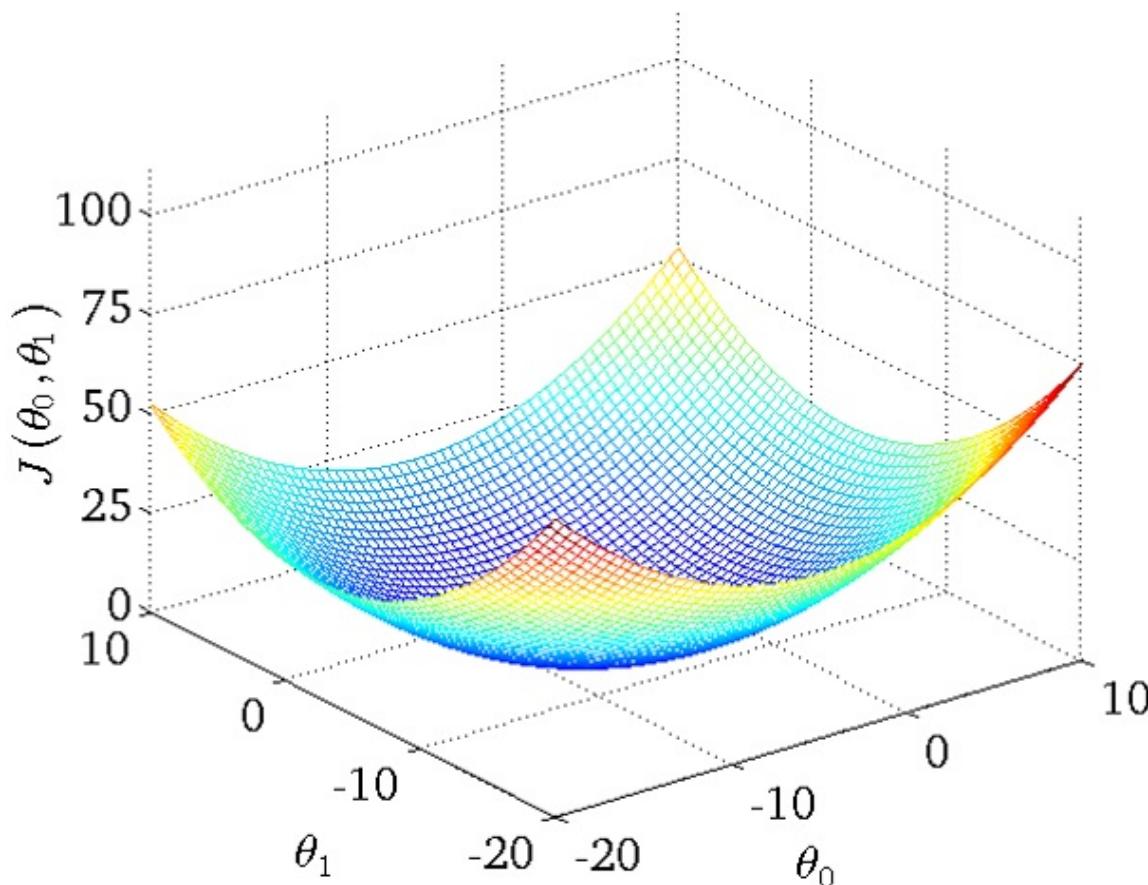
for i in range(len(J)):
    lbl = 'Hypothesis H = %0.1f * x' % theta[i]
    plt.scatter(x[i], J[i], linewidth=5, color=colors[i], label=lbl)

plt.legend(loc='best')
plt.title('Cost Function Graph')
plt.xlabel('Theta')
plt.ylabel('J (theta)')
plt.show()
```

গ্রাফ থেকে কী বুঝলাম? $\theta = 0.5$ এর জন্য কস্ট সবচেয়ে কম। মানে প্রেডিক্ষন সবচেয়ে বেটার যখন থিটার মান 0.5। এভাবে প্রতিটা মডেলের কস্ট ফাংশন থেকে আমরা ধারণা করতে পারি মডেলের পার্ফর্মেন্স কতটা ভাল।

যদি আমাদের মডেল $J(\theta_0, \theta_1) = \theta_0 + \theta_1 \times x$ হত

তাহলে সেটার প্রট হতে পারত এরকম,



আমরা অবশ্যে কস্ট ফাংশন সম্পর্কে অনেক কিছু জানতে পারলাম। এখন আমরা দেখব Cost Function Minimization Using **Gradient Descent**।

Gradient Descent অ্যালগরিদম

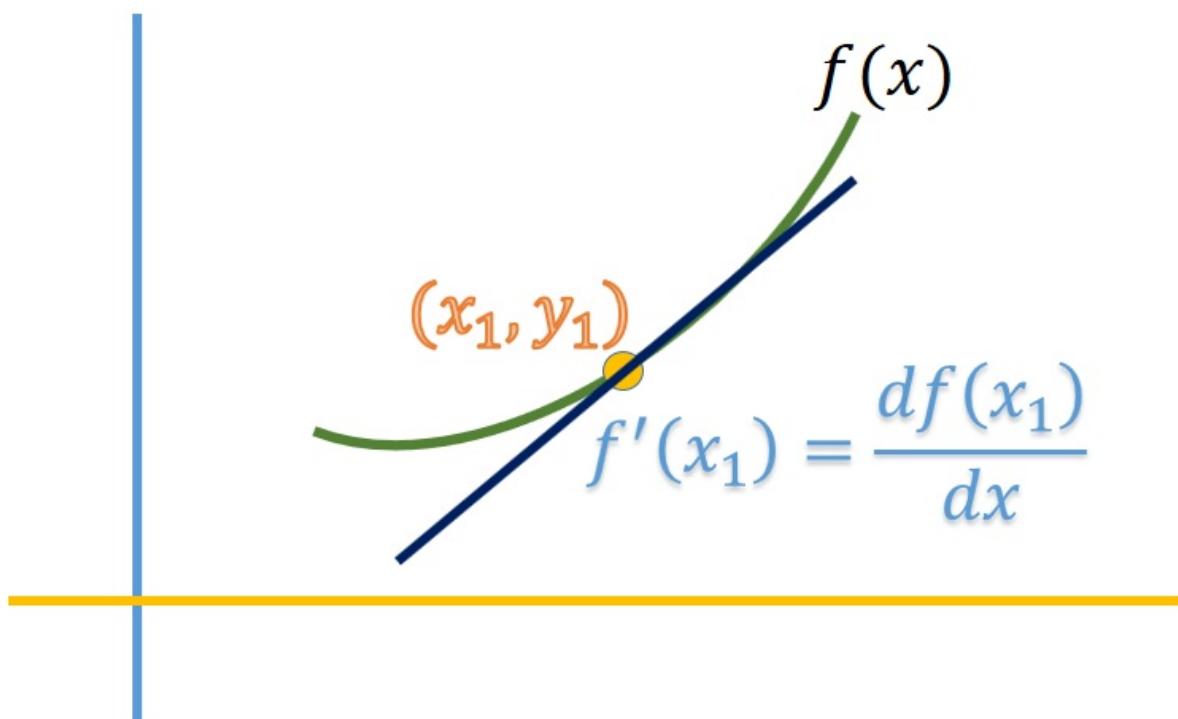
ক্যালকুলাস মনে আছে? ডিফারেনসিয়েশন? সেটাই আমাদের এখন কিছুটা কাজে আসবে। যদি মনে না থাকে তাহলে আগে একটু ডিফারেনসিয়েশন দেখা যাক।

Differentiation : Method for Calculating Slope at a specific point of a function

কোন বিলুতে কোন ফাংশনের ডেরিভেটিভ মানে হল সেই বিলুতে ও ফাংশনের স্পর্শকের ঢাল। ধরি, $y = f(x)$ যেকোন একটি ফাংশন, এখন আমরা তার (x_1, y_1) বিলুতে যে স্পর্শক, তার ঢাল (X অক্ষের সাথে বেখাটি কর ডিগ্রি কোণ উৎপন্ন করে) জানতে চাই। তাহলে আমরা $f(x)$ কে স্বাধীন চলক x এর সাপেক্ষে ডিফারেনসিয়েট

করব। ডিফারেনসিয়েট অপারেটর টা লেখে এইভাবে $\frac{dy}{dx}$ বা $\frac{df(x)}{dx}$ ।

নিচের ছবিটা দেখা যাক,



Slope বা ঢাল



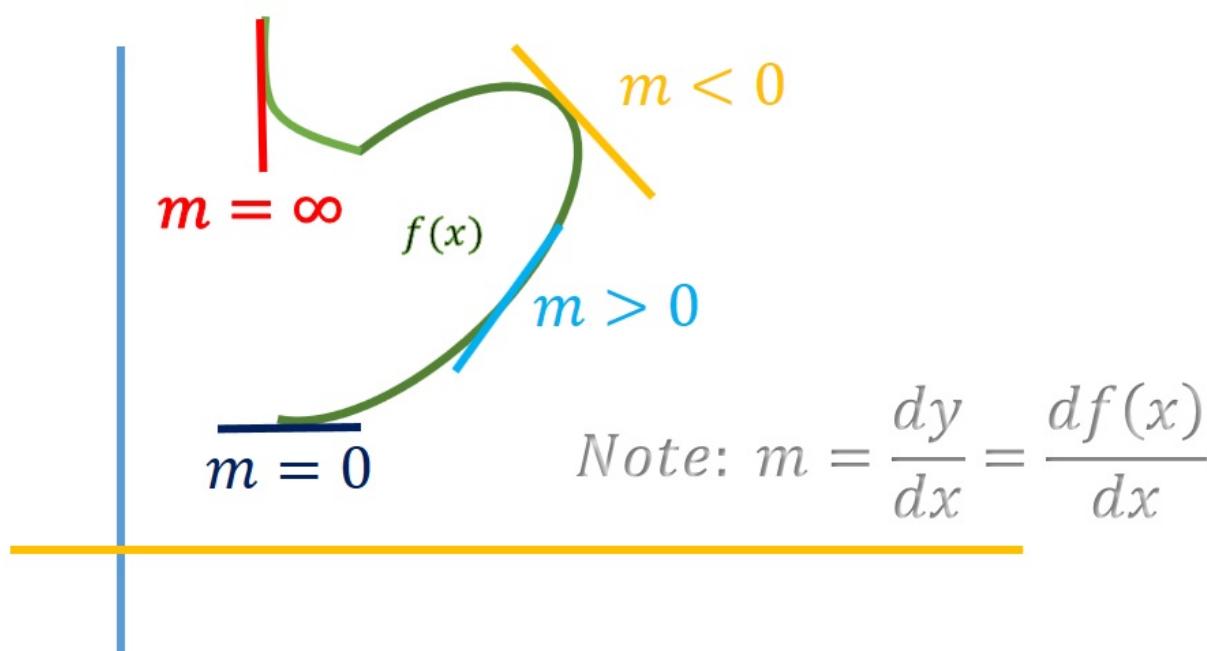
$$\text{ঢালের সূত্র হচ্ছে}, m = \frac{\Delta y}{\Delta x}$$

ঢালের মান চার ধরণের, নম-জিরো পজিটিভ, মেগেটিভ, জিরো এবং অসংজ্ঞায়িত। এই মানের ভিত্তিতে আমরা ঢালকে ক্লাসিফাই করতে পারি।

এই ঢাল চার ভাগ করা যায়,

- ধনাত্মক ঢাল (**Positive Slope**)
 - যে ঢাল X অক্ষের সাথে সূক্ষ্মকোণ উৎপন্ন করে সেটাকে ধনাত্মক ঢাল বলে। ধনাত্মক ঢাল আসলে বলে তার দিকে গেলে y এর মান বাঢ়বে।
- খণ্টাত্মক ঢাল (**Negative Slope**)
 - যে ঢাল X অক্ষের সাথে স্থুলকোণ উৎপন্ন করে সেটাকে খণ্টাত্মক ঢাল বলে। খণ্টাত্মক ঢাল বলে তার দিকে গেলে y এর মান কমবে।
- শূন্য ঢাল (**Zero Valued Slope**)
 - যে ঢাল X অক্ষের সাথে 0 ডিগ্রি কোণ উৎপন্ন করে সেটাকে শূন্য ঢাল বলে।
- অসংজ্ঞায়িত ঢাল (**Undefined Slope**)
 - যে ঢাল X অক্ষের সাথে 90 ডিগ্রি উৎপন্ন করে সেটাকে ধনাত্মক ঢাল বলে।

একনজরে ঢালগুলো,



Partial Derivative

আমাদের মূলত কাজে লাগবে পার্শ্বিয়াল ডেরিভিটিভ। একটা ফাংশন যে সব সময় একটা ড্যারিয়েবলের উপর ডিপেন্ডেন্ট থাকবে সেটা সত্য নয়। যেমন: $z = f(x, y) = x^2 + xy + y^2$ এই ফাংশনটার কথাই চিন্তা করা যাক, এখনে z ড্যারিয়েবলটি x, y দুইটার উপর নির্ভরশীল। তাই আমরা যদি x ও y দুইটার সাপেক্ষে z এর পরিবর্তন ট্র্যাক করতে চাই তাহলে একটা ডেরিভিটিভ দিয়ে হবে না।

$$z = x^2 + xy + y^2$$

$$\frac{\delta z}{\delta x} = 2x + y \quad \text{যখন } y \text{ ফ্রিবক}$$

$$\frac{\delta z}{\delta y} = 2y + x$$

যখন x প্রবক্ত

আমরা যদি θ_1 প্যারামিটার দিয়ে কস্ট ফাংশন ক্যালকুলেট করি তাহলে আমাদের সাধারণ ডেরিভেটিভ নিলেই হচ্ছে, কিন্তু যদি θ_0, θ_1 দুই কিংবা তার বেশি প্যারামিটার বিশিষ্ট কস্ট ফাংশন নেই তাহলে আমাদের অবশ্যই পার্শ্বিয়াল ডেরিভেটিভ নিতে হবে। আপাতত আমরা এক প্যারামিটার বিশিষ্ট কস্ট ফাংশন দিয়ে গ্রেডিয়েন্ট ডিসেন্ট বোঝার চেষ্টা করব।

প্রশ্ন আসতে পারে, এই ঢাল দিয়ে আমরা করব টা কী? আসলে ক্যালকুলাসের সামান্য(!) কনসেপ্ট দিয়ে আমরা বিলিওন বিলিওন সেকেন্ড বাঁচাতে পারি।

আমরা ডিফারেনসিয়েশন ও ঢালের কনসেপ্ট দিয়ে কস্ট মিনিমাইজ করার চেষ্টা করব। আর সেই চেষ্টার জন্য আমরা যে অ্যালগরিদম ব্যবহার করব সেটাই Gradient Descent।

গ্রেডিয়েন্ট ডিসেন্ট

অ্যালগরিদম

```
repeat until convergence {
     $\theta_j := \theta_j - \alpha \frac{\delta}{\delta \theta_j} J(\theta)$ 
}
```

ম্যাথমেটিক্যাল নোটেশন

মানে	ম্যাথ	প্রোগ্রামিং
x ও y সমান	$x = y$	$x == y$
y এর মান x এ অ্যাসাইন করা	$x := y$	$x = y$
x আপডেট উদাহরণ	$x := x + 1$	$x = x + 1$

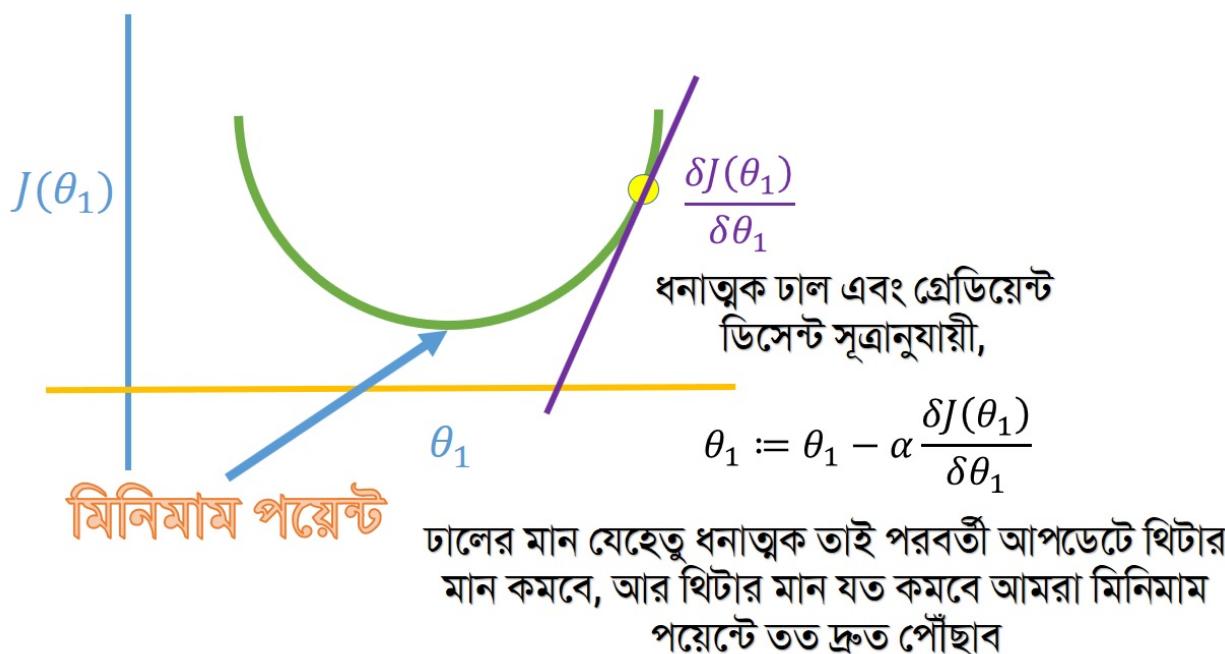
- তারমানে $:=$ এইটা দিয়ে বোঝানো হচ্ছে θ_j এর মান প্রতিবার আপডেট করতে হবে।
- এখানে α হল লার্নিং রেট (Learning Rate)

গ্রেডিয়েন্ট ডিসেন্ট ইনটুইশন

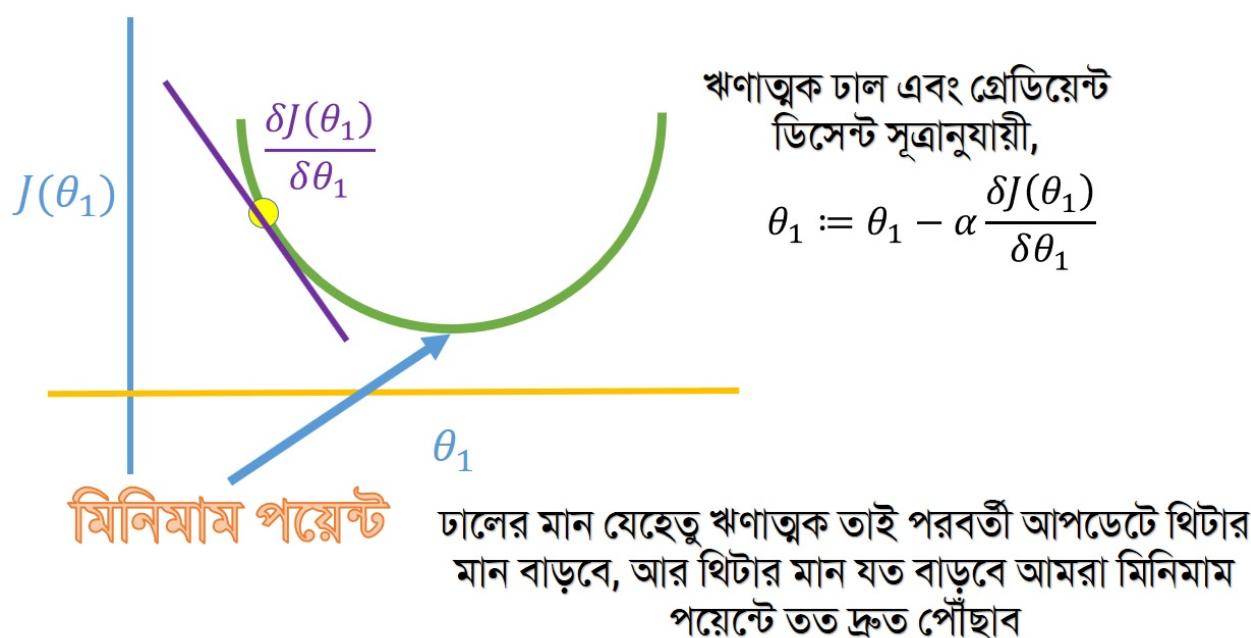
অ্যালগরিদম আসলে কী বলছে? আমরা আগেই জানি মেশিন লার্নিং মডেল ট্রেইনিং মানে হচ্ছে মডেলের ইন্টার্নাল প্যারামিটার গুলো এমন ভাবে সেট করা যাতে আমাদের প্রেডিকশন নির্ভুল হয়। আমরা কয়েকটা গ্রাফের মাধ্যমে বোঝাব চেষ্টা করি আসলে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদমের কাজটা কী।

ধরি আমাদের কস্ট ফাংশন $J(\theta_1)$

এইবার যেকোন একটা θ_1 এর মান ধরি, এবং সেই বিন্দুতে ডিফারেনসিয়েট করি। যদি ঢাল ধনাত্মক হয়, এর মানে ওঁদিকে গেলে $J(\theta_1)$ মান বাঢ়বে এবং উল্টা দিকে গেলে তার মান কমবে। নিচের ছবিটা দেখলেই বুঝা যাবে।



এইবার আমরা আরেকটা বিন্দু ধরি, যেটা কিনা লোকাল মিনিমাম এর বামে অবস্থান করে।



অর্থাৎ গ্রেডিয়েন্ট ডিসেন্ট সূত্রটি বলছে আমাদের কোন দিকে গেলে কষ্ট ফাংশনটা মিনিমাইজ হবে। এটা হল যখন একটা প্যারামিটার। এইরকম শত শত প্যারামিটারের সময় ডিজুয়ালাইজ করাটা সুবিধাজনক নয় তবে সব ক্ষেত্রে কাজটা ঠিক এইভাবেই হয়ে থাকে।

এই আপডেট ততক্ষণ চলতে থাকে যতক্ষণ না মিনিমাম পয়েন্টে পৌঁছাবেন। মিনিমাম পয়েন্টে অ্যালগরিদমটি

অটোমেটিক স্টপ হয়ে যাবে কারণ মিনিমাম পয়েন্টে $\frac{\delta J(\theta_1)}{\delta \theta_1} = 0$ আর গ্রেডিয়েন্ট অংশ যদি 0 হয় তাহলে আপডেটের কিছু থাকবে না।

এই পর্ব এই পর্যন্তে, পরবর্তী পর্বে আরেকদফা লিনিয়ার রিগ্রেশন, মাল্টি প্যারামিটারে গ্রেডিয়েন্ট ডিসেন্ট এবং ব্যাচ গ্রেডিয়েন্ট ডিসেন্ট সম্পর্কে জানতে পারব।

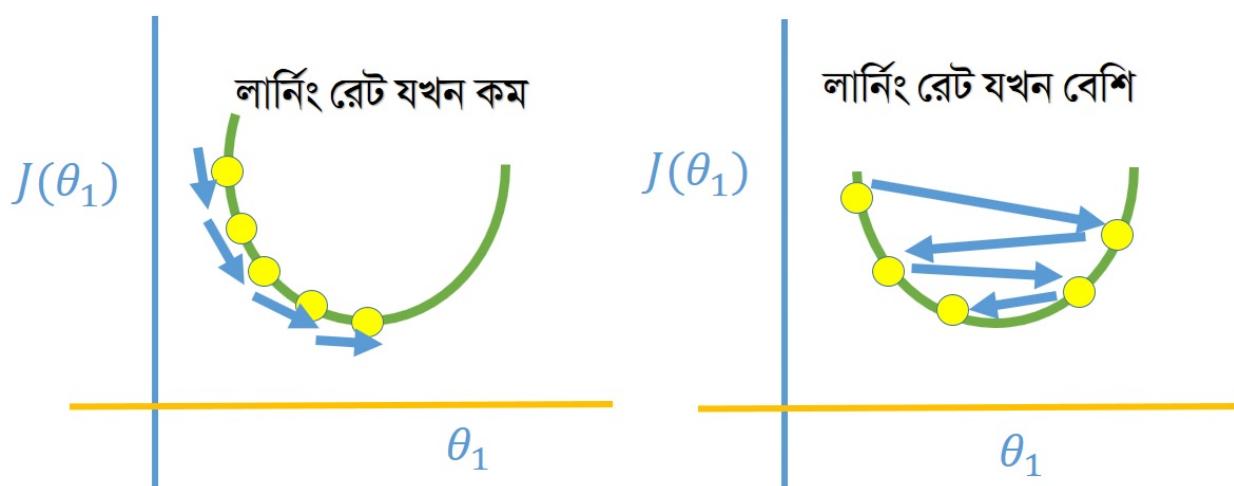
সচরাচর জিজ্ঞাস্য প্রশ্ন

লার্নিং রেট কী?

লার্নিং রেট বা α বলতে বুঝায় (ফিজিক্যাল মিনিং) কত দ্রুত কষ্ট ফাংশন লোকাল মিনিমামে কনভার্জ করতে চান। লার্নিং রেট কমালে θ_1 এর মান মিনিমামে কনভার্জ করতে সময় (ইটারেশন) বেশি নিবে মানে অনেকবার আপডেট হতে হবে। লার্নিং বাড়ালে আপডেট কম হবে। এই আলফা হতে হবে যেকোন পজিটিভ সংখ্যা।

লার্নিং রেট বাড়ালে বা কমালে কী ইফেক্ট সৃষ্টি হতে পারে?

মনে করুন, আপনার চোখে পটি বেঁধে একটা উচুনিচু ভূমিতে ছেড়ে দেওয়া হল। এবং বলা হল, আপনার কাজ হবে সবচেয়ে নিচু জায়গাটা বের করা। এখন যদি আপনি বড় বড় স্টেপে হাঁটেন তাহলে মিনিমাম পয়েন্ট এড়িয়ে যেতে পারেন, আবার ছোট ছোট স্টেপে হাঁটলে নিচু জায়গা বের করতে অনেক সময় লাগবে। এই যে স্টেপ নিচেন সেটাকে আমরা লার্নিং রেটের অ্যানালজি বলতে পারি।



স্টেপের সাথে সাথে লার্নিং রেট বাড়ানো/কমানোর দরকার আছে কী?

না নেই, কারণ মিনিমাম লোকাল পয়েন্টের দিকে আগাতে থাকলে অটোমেটিক গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদমের আপডেট স্টেপ করে যায়। তাই α এর মান যদি ফিক্সড থাকে তাহলেও সেটা মিনিমাম পয়েন্টে কনভার্জ করবে।

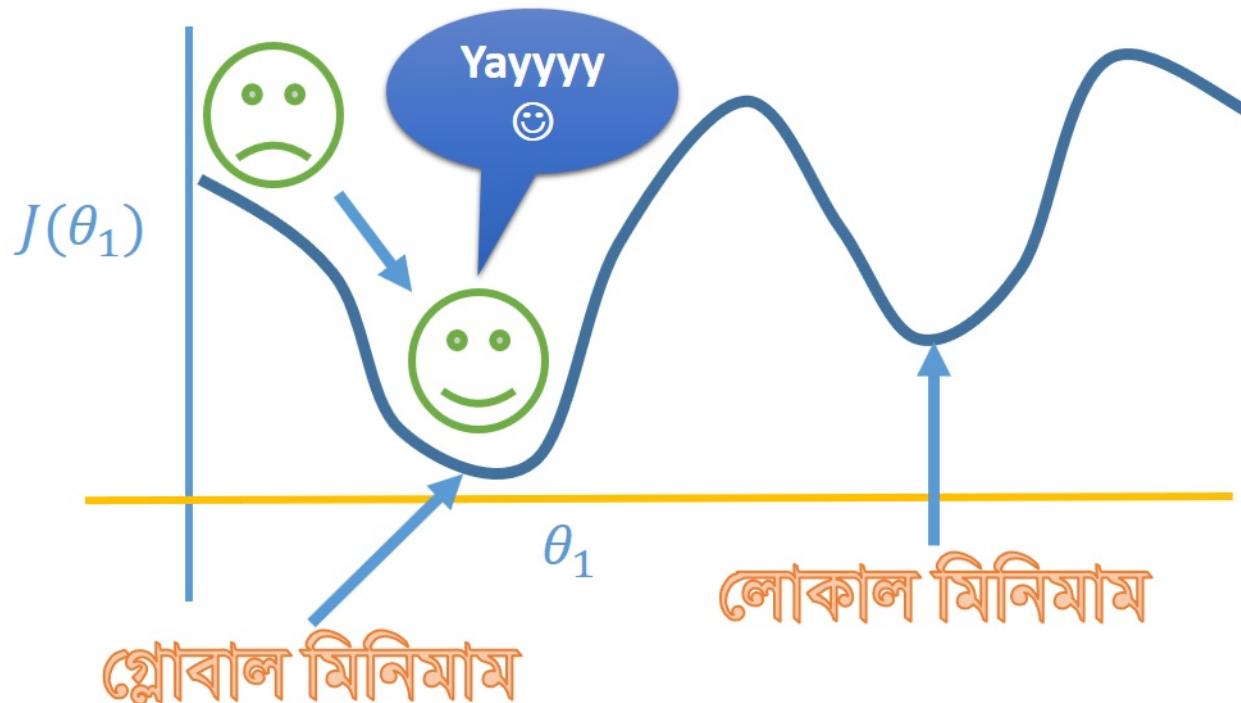
θ_1 এর মান বা সর্বোপরি প্যারামিটারগুলোর মান শুরুতে রংজ্যান্ডম নেওয়ার উদ্দেশ্য কী?

এই প্রশ্নের উত্তর অনেক বিশাল, রংজ্যান্ডম পয়েন্টে প্যারামিটার ইনিশিয়ালাইজেশনের মূল সুবিধা হচ্ছে ফ্রোবাল মিনিমাম বের করা। একই গ্রাফের লোকাল মিনিমাম বা ফ্রোবাল মিনিমাম থাকতে পারে। লোকাল মিনিমাম বলতে সেই পয়েন্ট কে বোঝানো হয় যেটা সামগ্রিক গ্রাফের মধ্যে তুলনামূলক নিম্নবিলু। আর ফ্রোবাল মিনিমাম হল পুরো গ্রাফের এমন একটা পয়েন্ট সেটাই সর্বনিম্ন বিলু।

আবার আমরা চোখে পট্টির উদাহরণে ব্যাক করি। ধরুন আপনাকে হেলিকপ্টারে করে এই পয়েন্টে ছেড়ে দিয়ে মিনিমাম পয়েন্ট বের করতে বলা হল। আপনি সোজা যেতে থাকলেন এবং লোকাল মিনিমাম বের করলেন। এখন যদি আপনাকে বার বার ওঁ পয়েন্টেই ছাড়ি এবং আপনি সোজাই যেতে থাকেন আপনি প্রত্যেকটা বার লোকাল মিনিমাম পয়েন্ট পেয়ে লাফালাফি শুরু করে দেবেন।



এবার আপনাকে রংজ্যান্ডমলি হেলিকপ্টার থেকে এই বিলুতে ছাড়া হল এবং এইবার আপনি আসলেই ফ্রোবাল পয়েন্টে যেতে পারবেন।



মাল্টিভ্যারিয়েবল লিনিয়ার রিগ্রেশন

গত পর্যুক্ত আমরা দেখেছিলাম সিঙ্গেল ভ্যারিয়েবল বিশিষ্ট সমস্যাগুলোতে কীভাবে লিনিয়ার মডেল ফিট করতে হয়। আজকে আমরা দেখব, সমস্যাটি যদি মাল্টি ভ্যারিয়েবল / কলাম / ফিচার বিশিষ্ট হয় তাহলে তার অ্যানালাইসিসটা কেমন হবে।

মাল্টিভ্যারিয়েবল বিশিষ্ট ডেটাসেট

কাজ শুরুর আগে ডেটাসেটটা একনজর দেখা যাক,

Size (feet^2)	Number of Bedrooms	Number of floors	Age of home (years)	Price (\$1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

লক্ষণীয়

লক্ষ করলে দেখা যাবে, আগের মত ইনপুট ভ্যারিয়েবল আর একটা নাই। বরং অনেকগুলো, তারমানে এখন আর আমরা ফিচার শুধু x ধরলেই হবে না। এখন আমাদের প্রতিটা কলাম ম্যাথেমেটিক্যাল নোটেশন দিয়ে আলাদা করতে হবে যেন আমরা বুঝতে পারি কোনটা আসলে কোন কলাম। এটা করার জন্য আমরা প্রতি কলামের জন্য x এর সাবস্ক্রিপ্ট দিয়ে কলাম নামার বসাব। সুপারস্ক্রিপ্টে রো (Row) ইন্ডেক্স বসবে এবং সাবস্ক্রিপ্টে বসবে কলাম (Column) ইন্ডেক্স।

উদাহরণ: (শুধু প্রথম Row এর জন্য)

$$\text{Size } (\text{feet}^2) = x_1^{(1)}$$

$$\text{Number of bedrooms} = x_2^{(1)}$$

$$\text{Number of floors} = x_3^{(1)}$$

$$\text{Age of home} = x_4^{(1)}$$

$$\text{Price} = y_1^{(1)}$$

তাহলে i তম ইনপুট ভ্যারিয়েবল হবে x_i এবং i তম আউটপুট ভ্যারিয়েবল হবে y_i

২য় উদাহরণ

আমরা যদি দ্বিতীয় সারির ইনপুট ভ্যারিয়েবলগুলোকে ম্যাট্রিক্সে সাজাতে চাই তাহলে সেটা হবে এইরকম, যেহেতু আমরা নির্দিষ্ট কোন Columnwise ভ্যারিয়েবল বিবেচনা করছি না, সবগুলো ভ্যারিয়েবল নিয়ে একটি ম্যাট্রিক্স তৈরি করেছি তাই আমাদের আলাদা করে সাবফিল্ড বসানোর মানে নেই।

$$X^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

এবং দ্বিতীয় সারির আউটপুট হবে,

$$Y^{(2)} = [232]$$

আপা করি তাহলে তৃতীয় ও চতুর্থ সারির ম্যাট্রিক্স নোটেশন কী হবে বুঝতে পেরেছেন। নোটেশন বোঝা শেষ, এবার আমরা সরাসরি চলে যাব মডেল বিন্দিংয়ে।

হাইপোথিসিস (Hypothesis)

আগের হাইপোথিসিস ছিল এটা,

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

এটা দিয়ে আমাদের এই মাল্টি ভ্যারিয়েবল সেটে কাজ করবে না। তাহলে উপায়? হ্যাঁ, উপায় আছে, সেটা হল প্রতিটা ভ্যারিয়েবলের আগে একটা করে নতুন প্যারামিটার গুণ করে দেওয়া।

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 \dots \quad (1)$$

এখন আমরা খিটার বিভিন্ন মান ধরে ভালমন্দ প্রেডিকশন করতে পারব, যেমন,

$$h_{\theta}(x) = 80 + 0.1x_1 + 0.01x_2 + 3x_3 - 2x_4 \dots \quad (2)$$

এই সমীকরণ (2) সিরিয়াসলি নেয়ার কিছু নাই, এটা চিঞ্চাভাবনাহীন উদাহরণ।

আবারও গাণিত

ভয়ের কিছু নেই, আমরা এখানে বেসিক ম্যাথেমেটিক্যাল নোটেশন নিয়েই আলোচনা করতে বসেছি। কারণ নোটেশনগুলো বুঝলে General Purpose Machine Learning এর খিওরি বুঝতে সমস্য হবে না, আমিও শর্টকাটে লিখতে পারব, আপনিও বুঝতে পারবেন।

হাইপোথিসিস মডিফিকেশন

আমরা সমীকরণ (1) এ মাল্টিভ্যারিয়েবল হাইপোথিসিস মডেলটা দেখতে পাচ্ছি। কথা হল, আমরা যদি সেটাকে ম্যাট্রিক্স আকারে সাজাতে চাই তাহলে বিশাল একটা সমস্যায় পড়ব। কারণ, হাইপোথিসিস এর প্যারামিটার শুরু হয়েছে θ_0 থেকে, কিন্তু ভ্যারিয়েবলের রো শুরু হয়েছে x_1 থেকে। তারমানে মডেল প্যারামিটারের সংখ্যা কলামের সংখ্যার চেয়ে বেশি। ম্যাট্রিক্সের যোগ বিয়োগ করতে হলে ডাইমেনশন সমান হতে হয়, ম্যাট্রিক্স অপারেশনগুলো কার্যকর করার জন্য তাই আমরা সমীকরণ (1) কে একটু মডিফাই করব।

আমরা সমীকরণ (1) কে লিখতে পারি এভাবে,

$$h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \cdots + \theta_n x_n \dots \quad (3)$$

যদি আমরা $x_0 = 1$ ধরি তাহলে সমীকরণ (2) এবং (3) এর মধ্যে পার্থক্য থাকবে না।

আমরা X ও θ কে যদি n সংখ্যক ভ্যারিয়েবলের ম্যাট্রিক্সে রাখতে চাই তাহলে আমরা লিখবো এভাবে,

$$X^{(i)} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

একই ভাবে খিটা প্যারামিটারগুলোকে আমরা যদি ম্যাট্রিক্স আকারে লিখি তাহলে দেখাবে এরকম,

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

কেন হাইপোথিসিস মডিফাই করা হল?

ম্যাট্রিক্স মাল্টিপ্লিকেশন : রুল নাষ্টার ১

দুইটা ম্যাট্রিক্স গুণ করার প্রথম শর্ত হল, প্রথম ম্যাট্রিক্সের কলাম সংখ্যা দ্বিতীয় ম্যাট্রিক্সের রো সংখ্যার সমান হতে হবে। আমরা যদি x_0 না বসাতাম তাহলে দুইটার ডাইমেনশন কখনই সমান হত না। অবশ্য এখনও আমরা দ্বিতীয় ম্যাট্রিক্স অর্থাৎ, θ কে ট্রান্সপোজ করি নাই, তাই একটু উলট পালট লাগতে পারে। ডাইমেনশন সমান করার আরেকটা সল্যুশন হতে পারত, আমরা যদি θ_0 উঠিয়ে দিতাম। কিন্তু প্যারামিটার উঠানো বুদ্ধিমানের কাজ নয়। আমাদের যদি একাঞ্জে নাই, তাই আমরা সেটার মান 0 বসিয়ে দিলেই হচ্ছে।

ম্যাট্রিক্স মাল্টিপ্লিকেশন উদাহরণ:

লিনিয়ার অ্যালজেব্রা মনে না থাকলে এটা একটা সামান্য আইওয়াশ হিসেবে নিতে পারেন, নিচের সমীকরণে,

$$Z = a_1 x_1 + a_2 x_2 + a_3 x_3$$

ধরি,

$$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

এবং

$$X = [x_1 \quad x_2 \quad x_3]$$

আমরা পুরো জিনিসটাকে তাহলে এভাবে ম্যাট্রিক্স আকারে লিখতে পারি,

$$Z = A \times X$$

তারমানে,

$$A \times X = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times [x_1 \quad x_2 \quad x_3] = a_1 x_1 + a_2 x_2 + a_3 x_3$$

কিন্তু,

উদাহরণে, একটা কলাম ও আরেকটা বো ম্যাট্রিক্স। কিন্তু আমরা যেসব ভ্যারিয়েবল নিয়ে কাজ করছি দুইটাই কলাম ম্যাট্রিক্স। তাই গুণ করার জন্য একটা কলাম ম্যাট্রিক্সকে বো ম্যাট্রিক্সে কনভার্ট করে নিতে পারি। এই কনভার্শনের নাম হল Transpose করা। ট্রান্সপোজ করা খুবই সহজ, ম্যাট্রিক্সের বো গুলিকে কলাম আকারে সাজালে কিংবা কলামগুলোকে বো আকারে সাজালেই হবে।

আমাদের এখানে মডিফাই করতে হবে যিটা ম্যাট্রিক্সকে, সুতরাং

$$\theta^T = [\theta_0 \quad \theta_1 \quad \theta_2 \quad \dots \quad \theta_n]$$

এখানে সুপারস্ক্রিপ্ট T দিয়ে ট্রান্সপোজ অপারেশন বুঝানো হয়েছে।

হাঈপোথিসিস ম্যাট্রিক্স নোটেশনে

$$h_0(x) = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n = \theta^T X$$

আশাকরি ভালমত বোরড হয়ে গেছেন, যাই হোক আর্টিফিশিয়াল ইন্টেলিজেন্স, ডেট সায়েন্স যেটাই হোক না কেন; লিনিয়ার অ্যালজেব্রা ছাড়া এক মূহর্তও চলে না। ইমেজ প্রসেসিং শেখার সময়ও একগাদা ম্যাট্রিক্স বেজড ম্যাথ নিয়ে ঘাঁটাঘাঁটি করা লাগবে।

মডিফাইড গ্রেডিয়েন্ট ডিসেন্ট

মাণ্টিভ্যারিয়েবল রিপ্রেশনের ক্ষেত্রে গ্রেডিয়েন্ট ডিসেন্টের অ্যালগরিদমও পরিবর্তিত হবে।

আগের অ্যালগরিদমটা ছিল,

```
repeat until convergence {
     $\theta_j := \theta_j - \alpha \frac{\delta}{\delta \theta_j} J(\theta)$ 
}
```

যেখানে,

$$\frac{\delta}{\delta \theta} J(\theta) = \frac{1}{m} \sum_{i=1}^m \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

যখন, $n = 1$

```
Repeat {
     $\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m \left( h_\theta(x^{(i)}) - y^{(i)} \right)$ 
     $\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m \left( h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}$ 
}
```

পরিবর্তিত সূত্র, যখন $n \geq 1$

```
Repeat {
     $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m \left( h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$ 
}
```

যেহেতু, একাধিক ভ্যারিয়েবলের জন্য,

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right) x_2^{(i)}$$

...

চলবে,

}

পরের পরে আমরা পাইথনে কোড লিখব।

প্র্যাক্টিক্যাল লিনিয়ার রিগ্রেশন : গ্রেডিয়েন্ট ডিসেন্ট

আমরা আজকের অধ্যায়ে লিনিয়ার রিগ্রেশন মডেল স্ক্যাচ থেকে তৈরি করা শিখব। তবে এটা করার আগে আপনার অবশ্যই Numpy সম্পর্কে ধারণা থাকতে হবে। না থাকলে [এই অধ্যায়টি পড়ে নিন](#)। তাহলে শুরু করা যাব।

ডেটাসেট

লিনিয়ার রিগ্রেশন মডেল বিন্দু করার জন্য আমি এখানে ব্যবহার করছি Andrew Ng এর Machine Learning কোর্সের লিনিয়ার রিগ্রেশন চ্যাপ্টারের ডেটাসেট। তবে সামান্য একটু পার্থক্য আছে। আমার দেওয়া ডেটাসেট কিছুটা পরিবর্তিত এবং পরিবর্তনটা হল প্রথম সারিতে শুধু দুইটা কলাম যোগ করে দিয়েছি। [ডেটাসেট দেখতে বা ডাউনলোড করতে এখানে ক্লিক করুন](#)

ডেটা ভিজুয়ালাইজেশন

সর্বপ্রথম আমরা যে কাজটি করব, সেটা হল আমার সংগৃহীত ডেটাসেট এর একটি স্ক্যাটার প্লট ড্র করা। আমি এখানে [Seaborn](#) লাইব্রেরি ব্যবহার করব। Seaborn লাইব্রেরিটি [matplotlib](#) এর উপর ভিত্তি করে তৈরি করা। ডেটা ভিজুয়ালাইজেশন সহজ করার জন্য অনেক ফিচার এতে বিল্ট-ইন আছে।

Seaborn ইন্সটলেশন

কম্পাউন্ড উইন্ডো বা টার্মিনালে নিচের কমান্ডটি রান করুন,

```
pip install seaborn
```

Seaborn ব্যবহার করে স্ক্যাটারপ্লট তৈরি করা

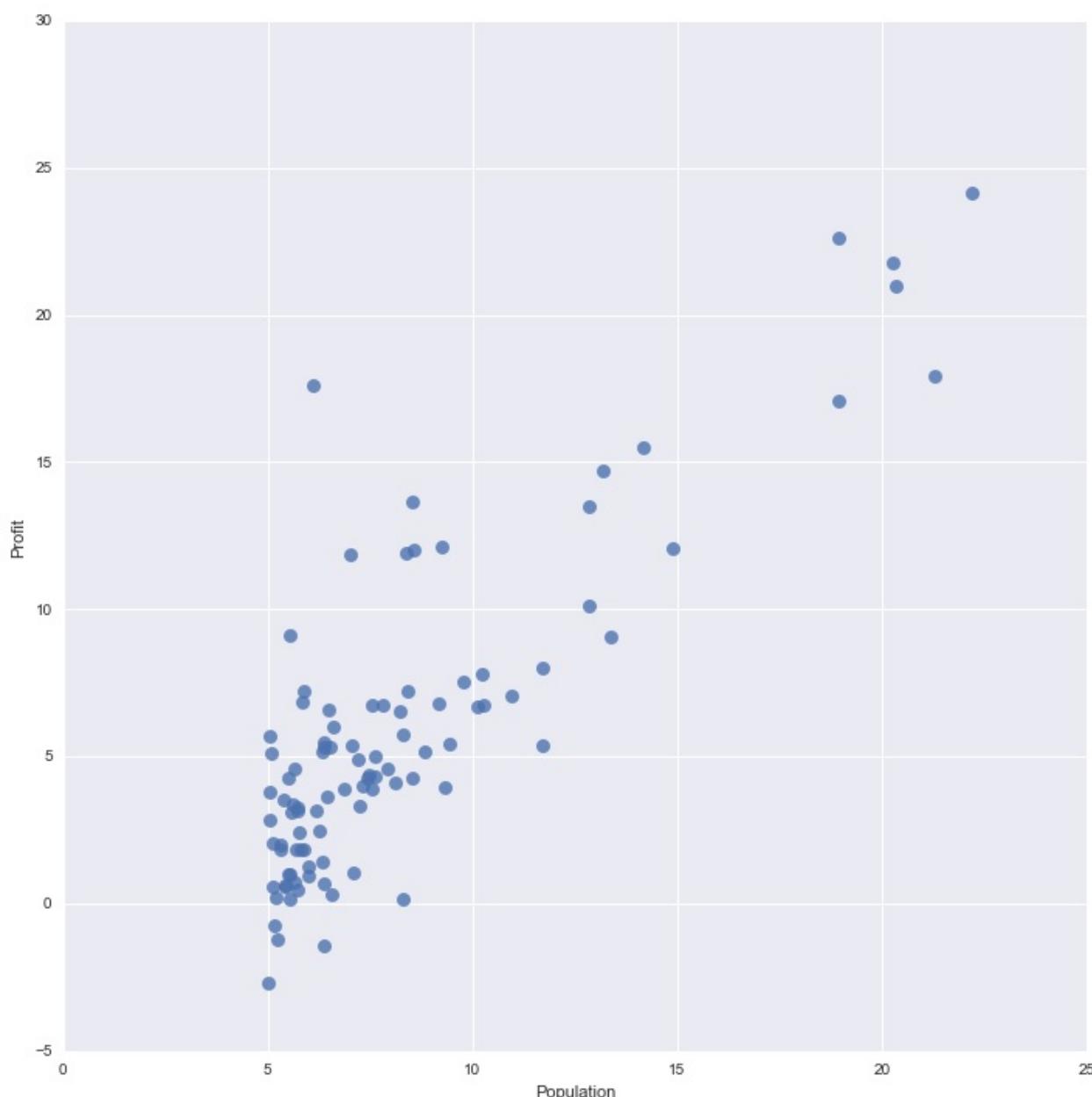
```
import csv
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import pandas as pd
import numpy as np

# Loading Dataset
with open('ex1data1.txt') as csvfile:
    population, profit = zip(*[(float(row['Population']), float(row['Profit'])) for row in csv.DictReader(csvfile)])

# Creating DataFrame
df = pd.DataFrame()
df['Population'] = population
df['Profit'] = profit

# Plotting using Seaborn
sns.lmplot(x="Population", y="Profit", data=df, fit_reg=False, scatter_kws={'s':45})
```

প্লট আউটপুট



ব্যাখ্যা

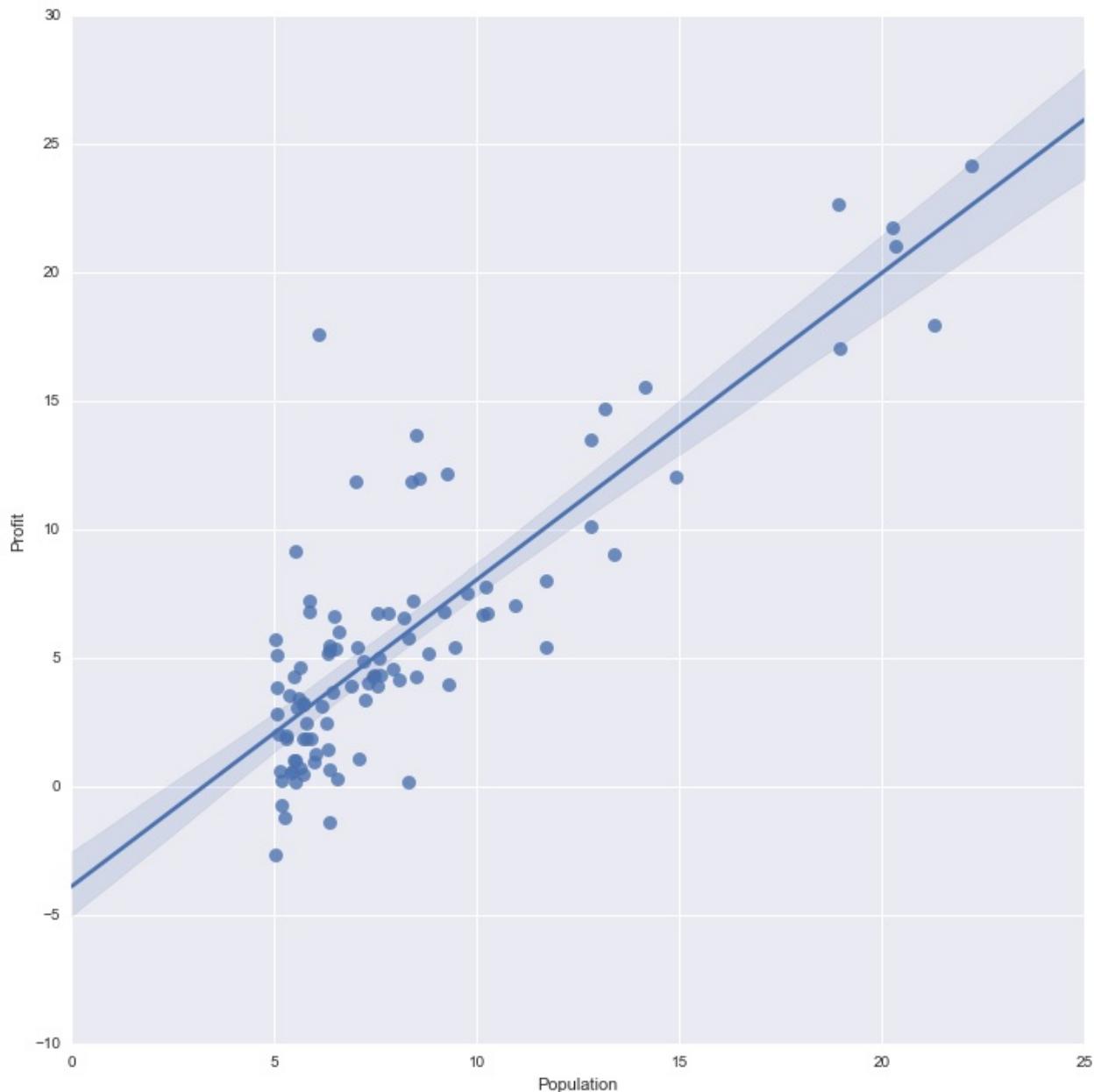
ডেটাসেট লোড ও ডেটাফ্রেম তৈরি

প্রথমেই আমি ডেটাসেট লোড করে দুইটা পাইথন লিস্টে ইনপুট ডেটা `Population` ও আউটপুট ডেটা / টাগেট / লেবেল নিলাম `Profit` লিস্টে। দুইটা লিস্ট দিয়ে একটা পার্সাস ডেটাফ্রেম তৈরি করলাম।

প্রটিং

`lmpplot` ফাংশন ব্যবহার করে স্ক্যাটারপট তৈরি করলাম যেখানে `x` ও `y` দিয়ে যথাক্রমে `x` ও `y` অক্ষে লেবেলিং করলাম এবং তৈরিকৃত ডেটাফ্রেমকে ডেটা হিসেবে পাস করলাম। `fit_reg` এর মান যদি `True` হত তাহলে গ্রাফটি দেখাত এমন, অর্থাৎ `Seaborn` একটা লিনিয়ার মডেলকে ফিট করে দেখাত। কিন্তু আমাদের মূল কাজটা সেটাই, গ্রেডিয়েন্ট ডিসেট অ্যালগরিদম ব্যবহারের মাধ্যমে করতে হবে। `scatter_kws={'s':45}` দিয়ে আমি স্ক্যাটার ডট গুলোর আকার পরিবর্তন করলাম।

`fit_reg=True` হলে প্রট যেমন দেখাত



কস্ট ক্যালকুলেশন ও গ্রেডিয়েন্ট ডিসেন্ট : ম্যাট্রিক্স অপারেশন

এই পর্যন্ত আমরা গ্রেডিয়েন্ট ডিসেন্ট ও কস্ট ক্যালকুলেশনের ব্যাপার স্যাপার দেখলাম। কিন্তু কোড এ হাত দেওয়ার আগে, যিরি টা আরেকটু ভালভাবে ঝালাই দেওয়া দরকার। কোড লেখার চাইতে গুরুত্বপূর্ণ বিষয় হল ডিজুয়ালাইজেশন। চলুন আমরা একটু ডিজুয়ালাইজ করে গ্রেডিয়েন্ট ডিসেন্ট প্রয়োগ করি।

কস্ট ক্যালকুলেশন

আমরা কষ্ট ক্যালকুলেশনের সূত্রানুযায়ী জানি,

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m \left(h_{\theta}(X^{(i)}) - Y^{(i)} \right)^2$$

এই ফরমুলা অ্যাপ্লাই করতে গেলে বুঝতেই পারছেন, লুপের ব্যবহার লাগবে। কিন্তু না, আমরা কাজটা Numpy ব্যবহার করে খুব সহজেই করতে পারি ম্যাট্রিক্স অপারেশনের মাধ্যমে। কোন নোটেশনের মানে কী সেটা আগের অধ্যয়গুলোতে বলা আছে। তাও আমি একটা সাধারণ উদাহরণের মাধ্যমে আবার দেখাই।

ধরি আমার ডেটাসেট এইটা,

i	আয় (X)	ব্যয় (Y)
1	10	5
2	20	10
3	30	15

যেখানে, $m = 3$

$$X^1 = 10, Y^1 = 5$$

$$X^2 = 20, Y^2 = 10$$

$$X^3 = 30, Y^3 = 15$$

লিনিয়ার রিগ্রেশন সূত্র,

$$h_{\theta}(X) = \theta \times X = \theta_0 \times X_i^0 + \theta_1 \times X_i^1 = \theta_0 + \theta_1 \times X$$

কিন্তু আমাদের যিটা এর ডাইমেনশন 2×1 অর্থাৎ, দুইটা সারি এবং একটা কলাম। ম্যাট্রিক্স আকারে,

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

এবং X এর ডাইমেনশন সিঙ্গেল ভ্যারিয়েল লিনিয়ার রিগ্রেশনের ক্ষেত্রে তাই আমরা একটি 1 এলিমেন্ট বিশিষ্ট কলাম যুক্ত করি। অর্থাৎ,

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix}$$

তাহলে আমাদের হাইপোথিসিস হবে প্রতিটা কলামের জন্য, $X \times \theta$

$$\begin{aligned}h_1(X) &= \theta_0 \times 1 + \theta_1 \times x_1 \\h_2(X) &= \theta_0 \times 1 + \theta_1 \times x_2 \\h_3(X) &= \theta_0 \times 1 + \theta_1 \times x_3\end{aligned}$$

ম্যাট্রিক্স আকারে লিখলে,

$$h_{\theta} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}$$

আউটপুট বা টার্গেট ম্যাট্রিক্স,

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

ম্যাট্রিক্স আকারে ক্যালকুলেটেড কস্ট সূত্র

$$\begin{aligned}J(\theta_0, \theta_1) &= \frac{1}{2 \times 3} \sum_{i=1}^3 \left(\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix} \times \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \right)^2 \\&= \frac{1}{6} \sum_{i=1}^3 \left(\begin{bmatrix} \theta_0 \times 1 + \theta_1 \times x_1 \\ \theta_0 \times 1 + \theta_1 \times x_2 \\ \theta_0 \times 1 + \theta_1 \times x_3 \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \right)^2 \\&= \frac{1}{6} \sum_{i=1}^3 \left(\begin{bmatrix} \theta_0 + \theta_1 \times x_1 - y_1 \\ \theta_0 + \theta_1 \times x_2 - y_2 \\ \theta_0 + \theta_1 \times x_3 - y_3 \end{bmatrix} \right)^2 \\&= \frac{1}{6} \{ (\theta_0 + \theta_1 x_1 - y_1)^2 + (\theta_0 + \theta_1 x_2 - y_2)^2 + (\theta_0 + \theta_1 x_3 - y_3)^2 \}\end{aligned}$$

বাড়ির কাজ

গ্রেডিয়েন্ট ডিসেন্ট ফরমুলা ম্যাট্রিক্স আকারে লিখুন।

এই ম্যাট্রিক্স ক্যালকুলেশনটাই আমরা পাইথনে লিখব। এসব কারণেই মেশিন লার্নিংয়ের ক্যালকুলেশন দ্রুত বুঝতে ও করতে লিনিয়ার অ্যালজেব্রার সলিড ফাউন্ডেশন দরকার। যে ভাল লিনিয়ার অ্যালজেব্রা ও ক্যালকুলাস বোঝে তার জন্য মেশিন লার্নিংয়ের অ্যালগরিদম অ্যাপ্লাই করা খুবই সহজ।

Numpy ব্যবহার করে কস্ট ক্যালকুলেশন ও গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম অ্যাপ্লাই করার পদ্ধতি

এখন আমরা Numpy ব্যবহার করে 97 অবজারভেশনের ডেটাসেট এর কস্ট ক্যালকুলেশন ও গ্রেডিয়েন্ট ডিসেন্ট এর ফাংশন লিখব।

পাইথনে কস্ট ক্যালকুলেশনের ফাংশন

```
# Here, X, y and theta are 2D numpy array
def computeCost(X, y, theta):
    # Getting number of observations
    m = len(y)

    # Getting hypothesis output
    hypothesis = X.dot(theta)

    # Computing loss
    loss = hypothesis - y

    # Computing cost
    cost = sum(loss**2)

    # Returning cost
    return (cost / (2 * m))
```

- কতটা অবজারভেশন আছে সেটা একটা `m` এ রাখলাম
- হাইপোথিসিস ড্যালু বের করলাম
- `loss` বের করলাম, যেটা কিনা আসল মান ও প্রেডিষ্টেড মানের বিয়োগফল
- `cost` বের করলাম, যেটা `loss` এর বর্গের যোগফল
- `average cost` রিটার্ন করলাম

পাইথনে গ্রেডিয়েন্ট ডিসেন্ট ক্যালকুলেশন ফাংশন

```

def gradientDescent(X, y, theta, alpha, iterations):
    cost = []
    m = len(y)
    for i in range(iterations):
        # Calculating Loss
        loss = X.dot(theta) - y
        # Calculating gradient
        gradient = X.T.dot(loss)
        # Updating theta
        theta = theta - (alpha / m) * gradient
        # Recording the costs
        cost.append(computeCost(X, y, theta))
        # Printing out
        print("Cost at iteration {0} : {1}".format(i, computeCost(X, y, theta)))
    return (theta, cost)

```

- আমরা একটা নির্দিষ্ট ইটারেশন রেঞ্জের মধ্যে প্যারামিটার আপডেট করব, অর্থাৎ, কস্ট একটা নির্দিষ্ট পরিমাণ কমে গেল সেটা আমাদের দেখার বিষয় না, একটা নির্দিষ্ট ইটারেশনে কতটুকু কস্ট কমে গেল। তাই আমরা ইটারেশন ফিঞ্চ করলাম। আবেকটা উপায় হতে পারে, একটা নির্দিষ্ট কস্ট হওয়ার আগ পর্যন্ত ইটারেশন চালিয়েই যাবে। কিন্তু সেটা অনেক ক্ষেত্রে বিপদ্জনক হতে পারে যেটা আমরা একটু পরেই দেখতে পারব।

তৈরিকৃত ফাংশন ব্যবহার করে এরর ক্যালকুলেশন প্লট বানানো

```

# Converting loaded dataset into numpy array
# Example:
# X = [[1, 10],
#       [1, 20],
#       [1, 30]]
#
X = np.concatenate((np.ones(len(population)).reshape(len(population), 1), population.reshape(len(population), 1)), axis=1)

# Example
# y = [[1],
#       [2],
#       [3]]
y = np.array(profit).reshape(len(profit), 1)

# Creating theta matrix , theta = [[0], [0]]
theta = np.zeros((2, 1))

# Learning rate
alpha = 0.1
# Iterations to be taken
iterations = 1500
# Updated theta and calculated cost
theta, cost = gradientDescent(X, y, theta, alpha, iterations)

```

আউটপুট

```

Cost at iteration 0 : 5.4441412681185035
Cost at iteration 1 : 5.409587207509947
Cost at iteration 2 : 5.376267659177092
Cost at iteration 3 : 5.344138517723003
Cost at iteration 4 : 5.313157253503435
Cost at iteration 5 : 5.2832828563299445
Cost at iteration 6 : 5.254475781184327
.....
.....
.....
Cost at iteration 23 : 177385094.9287188
Cost at iteration 24 : 9252868248.562147
Cost at iteration 25 : 482653703983.4108
Cost at iteration 26 : 25176474129882.656
Cost at iteration 27 : 1313270455375155.5
Cost at iteration 28 : 6.850360698103145e+16
Cost at iteration 29 : 3.573326537732157e+18

```

গ্রেডিয়েন্ট ডিসেন্ট ফরমুলা কাজ করছে না কেন?

গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদমের মূল কাজ কস্ট মিনিমাইজ করা, কিন্তু ইটারেশন 29 এই দেখুন কস্ট বেড়ে কত হয়েছে! এটা হওয়ার কারণ কী?

আসল কালপ্রিট : লার্নিং রেট

আমরা যদি লার্নিং রেট আরেকটু কমিয়ে কোড রান করি তাহলে,

```
# Creating theta matrix , theta = [[0], [0]]
theta = np.zeros((2, 1))

# Learning rate
alpha = 0.01
# Iterations to be taken
iterations = 1500
# Updated theta and calculated cost
theta, cost = gradientDescent(X, y, theta, alpha, iterations)
```

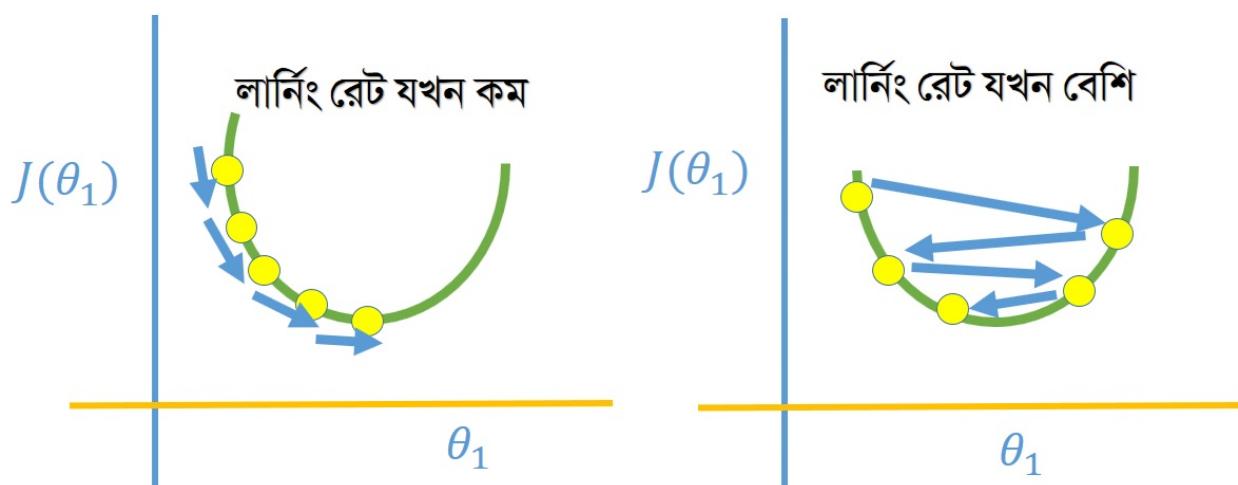
আউটপুট

```
Cost at iteration 0 : 6.737190464870004
Cost at iteration 1 : 5.9315935686049555
Cost at iteration 2 : 5.901154707081388
Cost at iteration 3 : 5.895228586444221
Cost at iteration 4 : 5.8900949431173295
Cost at iteration 5 : 5.885004158443647
Cost at iteration 6 : 5.879932480491418
Cost at iteration 7 : 5.874879094762575
Cost at iteration 8 : 5.869843911806385
....
....
....
Cost at iteration 1497 : 4.483434734017543
Cost at iteration 1498 : 4.483411453374869
Cost at iteration 1499 : 4.483388256587726
```

এবার দেখুন কস্ট আসলেই কমছে। অর্থাৎ আমাদের গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ঠিকঠাক কাজ করছে। লার্নিং রেট কমাতেই ওভারশটিং হচ্ছে না এবং গ্রেডিয়েন্ট হিল বেয়েই সে নিচে নামছে!

তাহলে কী সমস্যা হয়েছিল?

আপনি যদি [এই অধ্যায়টি](#) পড়ে থাকেন তাহলে বুঝবেন লার্নিং রেট বেশি হওয়ার কারণে সে মিনিমাম পয়েন্টে কনভার্জ না করে ওভারশট হওয়াতে শুধু উপরের দিকে যাচ্ছিল।

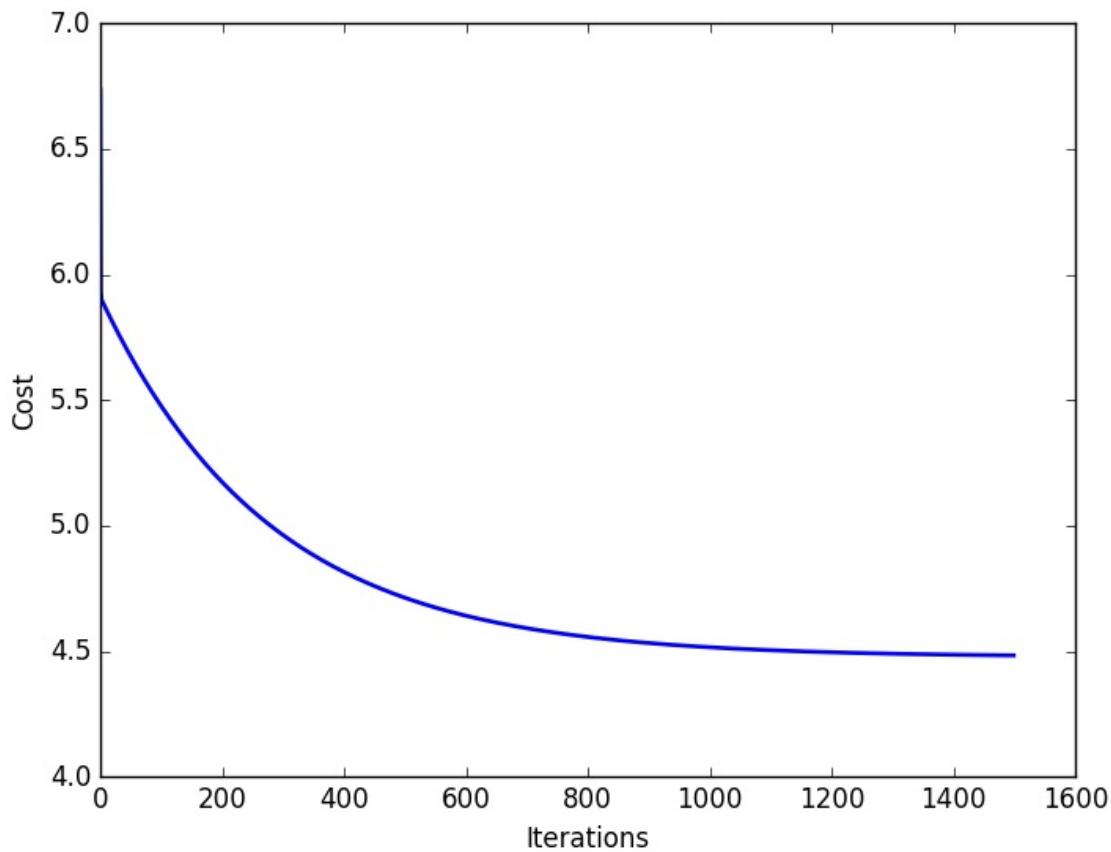


কস্ট বনাম ইটারেশন গ্রাফ

```
import matplotlib.pyplot as plt
plt.plot([i for i in range(1500)], cost, linewidth=1.9)
plt.xlabel("Iterations")
plt.ylabel('Cost')
plt.show()
```

আমরা যদি ইটারেশন vs কস্ট এর গ্রাফ প্লট করি তাহলে এটা দেখাবে এইরকম,

আউটপুট



মাল্টিভ্যারিয়েবল লিনিয়ার রিগ্রেশন

আমরা এতক্ষণ সিঙ্গেল ভ্যারিয়েবল লিনিয়ার রিগ্রেশন দেখলাম। মাল্টিভ্যারিয়েবলের ক্ষেত্রে কাজ পুরোপুরি একই, তবে কলাম সংখ্যা বেড়ে যাবে। তাতেও ম্যাট্রিক্স নোটেশন একই থাকবে।

পরবর্তী পর্বে গ্রেডিয়েন্ট ডিসেন্ট ম্যাট্রিক্স নোটেশনে প্রকাশ ও মাল্টিভ্যারিয়েবল লিনিয়ার রিগ্রেশনের জন্য মডেল তৈরি করব।

প্র্যাক্টিক্যাল মাল্টিভ্যারিয়েবল লিনিয়ার রিগ্রেশন : গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদমের রকমফের

গত পর্বে আমরা দেখেছিলাম সিঙ্গেল ভ্যারিয়েবল লিনিয়ার রিগ্রেশনের ফরমুলা ও পাইথনে ইম্প্রেমেন্টেশন। এই পর্বে
দেখব মাল্টিভ্যারিয়েবল বা মাল্টিফিচার বিশিষ্ট রিগ্রেশন সমস্যা কীভাবে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদমের মাধ্যমে
সল্ভ করে। তবে এখানে সিনথেটিক ডেটা দিয়ে সবকিছু আলোচনা করা হবে। সিনথেটিক ডেটা হল কোন
ম্যাথেমেটিক্যাল ফরমুলা ব্যবহার করে জেনারেট করা ডেটা। আজকের আলোচ্য কল্টেন্ট দেখা যাব।

গ্রেডিয়েন্ট ডিসেন্টের প্রকারভেদ জানা খুবই দরকারী। প্রায় সময়ই ফাংশন অপ্টিমাইজেশনের ক্ষেত্রে এই কথাগুলো
বেশি পরিমাণে ব্যবহৃত হয়। সাধারণত লিনিয়ার রিগ্রেশনের ক্ষেত্রে এদের ব্যবহার দেখানো হয় না, কিন্তু সিম্পল লিনিয়ার
মডেল দিয়েই অ্যালগরিদমগুলো ডাল বোঝা যায় তাই এখানে আলোচনা করা হল।

আলোচ্য বিষয়বস্তু:

- গ্রেডিয়েন্ট ডিসেন্ট ফরমুলা ডিরাইভ করা
- সিনথেটিক ডেটা প্রস্তুত করা
- ব্যাচ গ্রেডিয়েন্ট ডিসেন্ট (Batch Gradient Descent), স্টোক্যাস্টিক গ্রেডিয়েন্ট ডিসেন্ট (Stochastic Gradient Descent) ও মিনি-ব্যাচ গ্রেডিয়েন্ট ডিসেন্ট (Mini-batch Gradient Descent) এর সুবিধা-
অসুবিধা ও পাইথনে ইম্প্রেমেন্টেশন

লিনিয়ার রিগ্রেশনের জন্য গ্রেডিয়েন্ট ডিসেন্ট

গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম বলে প্যারামিটারের ভ্যালু আপডেট করতে হবে এভাবে,

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

যেখানে,

- θ হল প্যারামিটার
- j দিয়ে বুঝাচ্ছে কততম প্যারামিটার
- α হল লার্নিং রেট
- $J(\theta)$ দিয়ে কষ্ট ফাংশন বুঝানো হচ্ছে

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

আবার যেখানে

- m হচ্ছে বো সংখ্যা বা ডেটা কতগুলো আছে
- h_{θ} হচ্ছে হাইপোথিসিস ফাংশন যেটা আমরা পরে ডিফাইন করব
- $y^{(i)}$ হল ডেটাসেট এ দেয়া আউটপুট ভ্যালু
- $x^{(i)}$ হল ইনপুট ডেটা

গ্রেডিয়েন্ট ডিসেন্ট ফরমুলেশন

আগে দেখানো হয়েছিল কীভাবে ম্যাট্রিক্সে কস্ট ক্যালকুলেট করে কিন্তু গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদমে কস্ট ফাংশনকে ডেরিভেটিভ করলে কী আসে সেটা দেখানো হয় নি। শুরু করা যাক।

লিনিয়ার রিপ্রেজেন্টেশনের হাইপোথিসিস সাধারণত এরকম হয়,

$$h_{\theta}(X) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2$$

যেখানে,

$$X = x = \begin{bmatrix} x_0 & = 1 \\ x_1 \\ x_2 \end{bmatrix}$$

যদি ইনপুটের ফিচার বা কলাম সংখ্যা হয় দুইটা। ডেটাসেট এ দুইটা ফিচার থাকলেও আমরা একস্ট্রা একটা কলাম 1 সেট করে নেব। ম্যাট্রিক্সে ফরমুলেট করার জন্য এই 1 এর গুরুত্ব অপরিসীম। (সাইড নোট: ইনপুট ভ্যারিয়েবলে মাঝে মাঝে X বা x হিসেবে লিখব, দুইটার মানে একই)

আমরা হাইপোথিসিস ফাংশনকে এভাবে আরও সংক্ষিপ্ত আকারে লিখতে পারি,

$$h_{\theta}(X) = \sum_{i=0}^n \theta_i x_i$$

ম্যাট্রিক্স আকারে,

$$h_{\theta}(X) = \theta^T X$$

বিখ্যাত ডেটাসেটটা আবার আনি,

বাড়ির আকার (x_1) (sq-ft)	ঘর সংখ্যা (x_2)	বাড়ির দাম y (lac)
1200	5	120
1300	6	125
1400	7	130

এখনে, $m = 3$ কারণ Row মাত্র তিনটা। এবং ফিচার দুইটা x_1, x_2 । যেহেতু x_0 সহ তিনটা তাই $n = 3$ ডেটাসেট পাওয়ার পরে আমরা একস্ট্রা একটা ফিচার কলাম জুড়ে দিলে হবে এরকম,

x_0	বাড়ির আকার (x_1) (sq-ft)	ঘর সংখ্যা (x_2)	বাড়ির দাম y (lac)
1	1200	5	120
1	1300	6	125
1	1400	7	130

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

এই ফরমুলাতে আমাদের আগে বের করতে হবে, $\frac{\partial J(\theta)}{\partial \theta_j}$ এটার মান কত। এটার মান বের করতে পারলে তারপর অ্যালগরিদমে বসিয়ে দিলেই হবে।

$$\begin{aligned} \frac{\partial J(\theta)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} \left[\frac{1}{2m} (h_{\theta}(x) - y)^2 \right] \\ &= 2 \times \frac{1}{2m} (h_{\theta}(x) - y) \times \frac{\partial}{\partial \theta_j} (h_{\theta}(x) - y) \\ &= \frac{1}{m} \times (h_{\theta}(x) - y) \times \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^n \theta_i x_i - y \right) \\ &= \frac{1}{m} \times (h_{\theta}(x) - y) \times \left[\frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^n \theta_i x_i \right) - \frac{\partial y}{\partial \theta_j} \right] \\ &= \frac{1}{m} \times (h_{\theta}(x) - y) \times \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^n \theta_i x_i \right) \end{aligned}$$

এখনে $\frac{\partial y}{\partial \theta_j} = 0$ কারণ আউটপুট মানগুলো ভ্যারিয়েবল না, ফ্র্মুলা। তাই তাদের ডেরিভেটিভ নিলে ফলাফল আসবে শূন্য।

শেষের লাইনে একটি সমস্যা চলে এসেছে, সেটা হল হাইপোথিসিস ফাংশনের ডেরিভেটিভ কীভাবে বের করব। সেটা বের করার জন্য এক্সপ্রেশন আরও ডেঙ্গে লেখা যাক।

$$h_{\theta}(X) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2$$

এই ফাংশনের পার্শ্ব্যাল ডেরিভেটিভ আমরা θ_0 এর সাপেক্ষে নিলে, (মনে আছে তো? পার্শ্ব্যাল ডেরিভেটিভ যে ভ্যারিয়েবলের সাপেক্ষে নিব, সেটা বাদে সবাইকে ফ্র্মুলা হিসেবে গণনা করতে হবে।)

$$\frac{\partial h_{\theta}(x)}{\partial \theta_0} = \left(x_0 \times \frac{\partial \theta_0}{\partial \theta_0} + \theta_0 \times \frac{\partial x_0}{\partial \theta_0} \right) + \frac{\partial}{\partial \theta_0} (\theta_1 x_1) + \frac{\partial}{\partial \theta_0} (\theta_2 x_2)$$

$x_0, \theta_1 x_1, \theta_2 x_2$ সবগুলোই ফ্র্যাক্ষন এর কাছে, তাই

$$\begin{aligned}\frac{\partial x_0}{\partial \theta_0} &= 0 \\ \frac{\partial}{\partial \theta_0} (\theta_2 x_2) &= 0 \\ \frac{\partial}{\partial \theta_0} (\theta_1 x_1) &= 0\end{aligned}$$

সুতরাং,

$$\frac{\partial h_{\theta}(x)}{\partial \theta_0} = x_0 \times 1 = x_0$$

একইভাবে,

$$\begin{aligned}\frac{\partial h_{\theta}(x)}{\partial \theta_1} &= x_1 \\ \frac{\partial h_{\theta}(x)}{\partial \theta_2} &= x_2\end{aligned}$$

জেনারেল ফর্মে,

$$\frac{\partial h_{\theta}(x)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^n \theta_i x_i \right) = x_j$$

এই সূত্র এবার কষ্ট ফাংশনের ডেরিভেটিভে বসিয়ে পাওয়া যায়,

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \times (h_{\theta}(x) - y) \times x_j$$

সিঙ্গেল প্যারামিটার আপডেটের গ্রেডিয়েন্ট ডিসেন্ট ফরমুলা সম্পূর্ণরূপে লিখলে,

$$\theta_j := \theta_j - \frac{\alpha}{m} \times (h_{\theta}(x^{(i)}) - y^{(i)}) \times x_j^{(i)}$$

অথবা এভাবেও লেখা যায় যদি আমি আউটপুট আগে বসাই,

$$\theta_j := \theta_j + \frac{\alpha}{m} \times (y^{(i)} - h_{\theta}(x^{(i)})) \times x_j^{(i)}$$

নোটেশনে যাতে সমস্যা না হয় তাই আরেকবার লিয়ার করা যাক,

- $x^{(i)}$ কিন্তু ডেটাসেট এর i^{th} row। অর্থাৎ উপরের উদাহরণ অনুযায়ী, 2^{nd} রো হল,

$$x^{(2)} = \begin{bmatrix} x_0^{(2)} = 1 \\ x_1^{(2)} = 1300 \\ x_2^{(2)} = 6 \end{bmatrix}$$

- $x_j^{(i)}$ হল ডেটাসেট এর i^{th} রো এর j^{th} ফিচার। অর্থাৎ কিনা, $x_2^{(2)} = 6$

গ্রেডিয়েন্ট ডিসেট ইম্প্রেভেশনে এই জিনিসটা খুবই গুরুত্বপূর্ণ। এই পর্যন্ত বুঝে থাকলে ইম্প্রেভেট করা বা কোড বোঝা কোন ব্যাপারই না।

এই অ্যালগরিদম আমি তো কোন মডেলের উপর অ্যাপ্লাই করব, আর যে মডেল বিন্দু করব সেটা নির্ভর করবে ডেটাসেটের উপর। একটা নির্দিষ্ট কারণে আমি বাইরের ডেটাসেট ব্যবহার না করে সিনথেটিক্যালি ডেটাসেট প্রস্তুত করে সেটাৰ উপর অ্যালগরিদম অ্যাপ্লাই করব।

সিনথেটিক ডেটা প্রস্তুতকরণ

ডেটা বিন্দু করার জন্য আমি একটা সিম্পল ফরমুলা ব্যবহার করব। সেটা হল এরকম,

$$y = 5 + 2 \times x_1 + 3 \times x_2$$

যেখানে x_1, x_2 যেকোন ভ্যালু হতে পারে।

```
import numpy as np

def make_fake_data(x1, x2):
    # y = 5+2*x1+3*x2
    return float(5 + 2 * x1 + 3 * x2)

# first feature of input data, col - 1
feature_1 = np.array([float(i) for i in range(1, 21)])
# Second feature of input data, col - 2
feature_2 = np.array([float(i) for i in range(21, 41)])

# Output
y = np.array([
    make_fake_data(f1, f2)
    for f1, f2 in
    zip(feature_1, feature_2)
])
```

বলাই বাহল্য, এখানে x_1, x_2 হল ইনপুট ডেটার ফিচার বা কলাম। এই ফরমুলা দিয়ে বানানো ডেটাসেট হবে,

x_1	x_2	y
1	21	70
2	22	75
3	23	80
4	24	85
...
20	40	165

যেহেতু ইনপুট ফিচার দুইটা সেক্ষেত্রে আমাদের হাইপোথিসিস মডেল হবে,

$$h_{\theta}(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

আমরা একে ম্যাট্রিক্সে ফরমুলেট না করে প্রথমে প্যারামিটারওয়াইজ ইটারেচিভ পদ্ধতিতে সল্ভ করব এবং পরে ম্যাট্রিক্স আকারে কিভাবে সল্ভ করে সেটা দেখব। তাই আমরা $x_0 = 1$ ফিচার ব্যবহার করিনি।

```
def predict(_theta_0, _theta_1, _theta_2, x1, x2):
    return _theta_0 + _theta_1 * x1 + _theta_2 * x2
```

গ্রেট, আমাদের হাইপোথিসিস ফাংশন আছে, ইনপুট ডেটা আর আউটপুট ডেটা আছে, আমাদের এখন অজানা $\theta_0, \theta_1, \theta_2$ এর মান বের করতে হবে শুধু।

কস্ট ক্যালকুলেশন ফাংশন:

প্রতি ইটারেশনে কস্ট কত সেটা ক্যালকুলেট করলে আমরা গ্রাফ প্লট করে দেখতে পারব এবং আসলেই কমছে কিনা। চিরাচরিত কস্ট ফাংশন,

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

এটাকে এবার পাইথন ফাংশন আকারে লিখব,

```

def computeCost(t0, t1, t2, f1, f2, y):
    # Getting number of data
    m = float(len(y))
    loss = []
    # Iterating over all of the data
    for i in range(len(y)):
        # Getting prediction using the parameter [t0, t1, t2]
        h = predict(t0, t1, t2, f1[i], f2[i])
        # Adding the losses to the list
        loss.append((h - y[i])**2)

    return (sum(loss) / (2 * m))

```

কস্ট বনাম ইটারেশন প্লট করার ফাংশন

এটা আমাদের কাজে আসবে তাই ফাংশন বানিয়ে নেয়াই ভাল,

```

def plot_cost_vs_iteration(costs):
    plt.plot([i for i in range(len(costs))], costs)
    plt.title("Cost vs Iteration")
    plt.xlabel("Iteration")
    plt.ylabel("Cost")
    plt.show()

```

গ্রেডিয়েন্ট ডিসেন্টের সাধারণত তিনরকম প্রকারভেদ দেখা যায়

- ব্যাচ গ্রেডিয়েন্ট ডিসেন্ট (Batch Gradient Descent)
- স্টোক্যাস্টিক গ্রেডিয়েন্ট ডিসেন্ট (Stochastic Gradient Descent)
- মিনি-ব্যাচ গ্রেডিয়েন্ট ডিসেন্ট (Mini-Batch Gradient Descent)

ব্যাচ গ্রেডিয়েন্ট ডিসেন্ট

প্রথমে $\theta_0, \theta_1, \theta_2$ এর মান বের করব ব্যাচ গ্রেডিয়েন্ট ডিসেন্ট দিয়ে, তারপর বাকি অ্যালগরিদম দিয়েও বের করব।

ব্যাচ গ্রেডিয়েন্ট ডিসেন্টের ফরমুলা,

$$\text{Repeat Until Convergence} \{$$

$$\theta_j := \theta_j + \alpha \sum_i^m \left(y^{(i)} - h_{\theta}(x^{(i)}) \right) x_j^{(i)} \quad (\text{for every } j)$$

$$\}$$

```

# Parameters to be updated
_theta_0 = 0.0
_theta_1 = 0.0
_theta_2 = 0.0

# Data Length
m = float(len(y))

# Epoch [No. of iterations]
epoch = 200

# Learning Rate
alpha = 0.001

# Costs
costs = []

# Batch Gradient Descent
for i in range(epoch):

    _theta_temp_0 = _theta_0 + (alpha / m) * sum([(y[k] - predict(_theta_0, _theta_1,
_theta_2, feature_1[k], feature_2[k])) *
for k in range(len(y))
])

    _theta_temp_1 = _theta_1 + (alpha / m) * sum([(y[k] - predict(_theta_0, _theta_1,
_theta_2, feature_1[k], feature_2[k])) * feature_1[k]
for k in range(len(y))
])

    _theta_temp_2 = _theta_2 + (alpha / m) * sum([(y[k] - predict(_theta_0, _theta_1,
_theta_2, feature_1[k], feature_2[k])) * feature_2[k]
for k in range(len(y))
])

    _theta_0 = _theta_temp_0
    _theta_1 = _theta_temp_1
    _theta_2 = _theta_temp_2

    # Calculating cost
    cost = computeCost(_theta_0, _theta_1, _theta_2, feature_1, feature_2, y)

    # Saving it to the list for future use
    costs.append(cost)

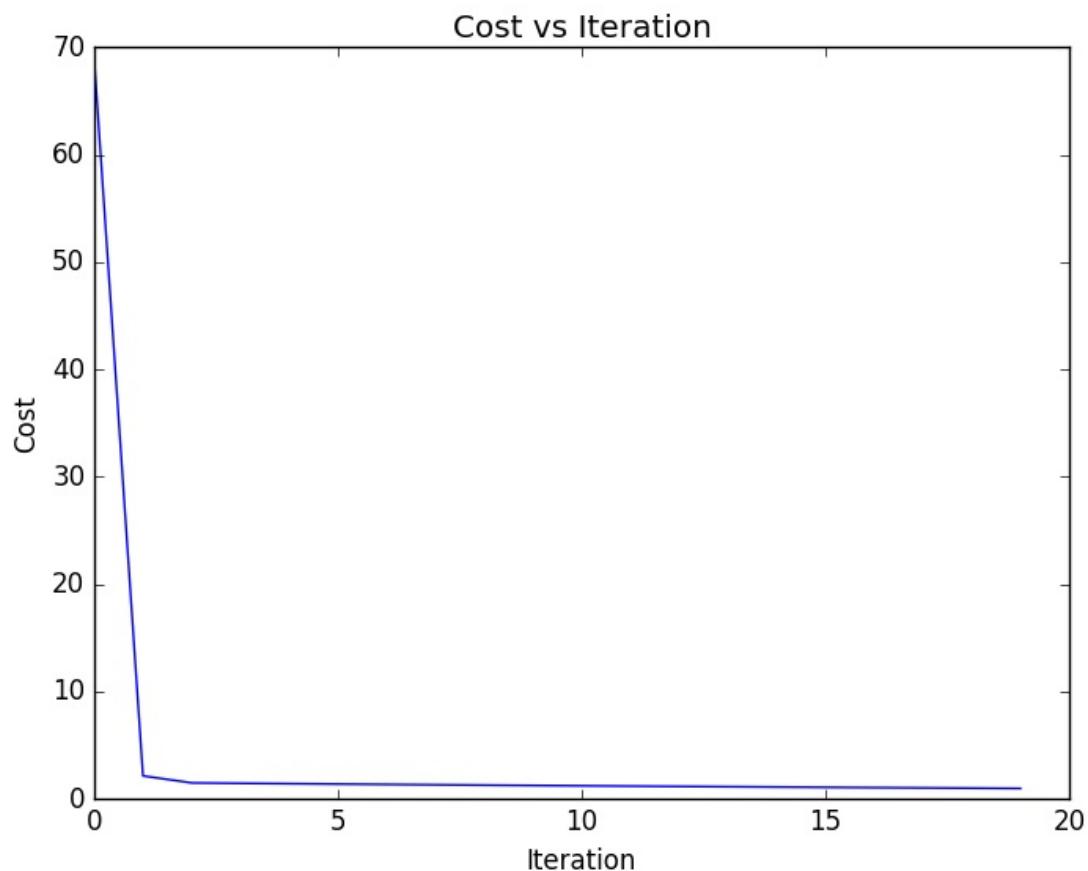
    # Printing cost after each epoch
    print("Cost: {}".format(cost))

# Plotting Cost vs Iteration Graph
plot_cost_vs_iteration(costs)

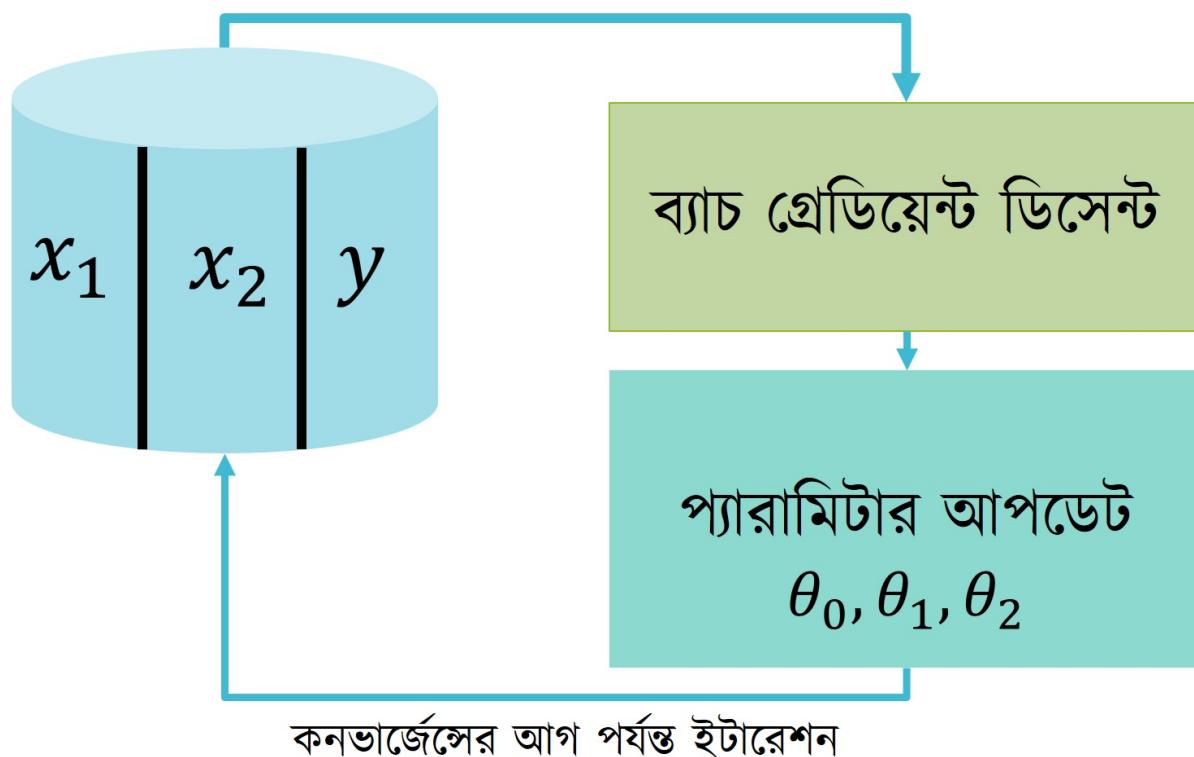
```

এখানে কনভার্জেন্স শর্ত না বসিয়ে ইটারেশন দিয়ে লিমিট করে দেয়া হল।

কষ্ট বনাম ইটারেশন প্লট



ব্যাচ গ্রেডিয়েন্ট ডিসেন্টে প্যারামিটার আপডেট



ব্যাচ গ্রেডিয়েন্টের মূল অসুবিধা হল প্রতিবার প্যারামিটার ভ্যালু আপডেটের সময় বার বার সম্পূর্ণ ডেটাসেট ইটারেট করতে হয় এবং প্রতি প্যারামিটারের জন্য।

এই অসুবিধা দূর করার জন্য আরেকটি অ্যালগরিদম প্রায়ই ব্যবহার করা হয় যার নাম Stochastic Gradient Descent।

স্টোক্যাস্টিক গ্রেডিয়েন্ট ডিসেন্ট

Stochastic এর মানে হল কোন কিছু বজ্যান্তমলি ডিটারমাইন করা। এখানে আমরা কস্ট ফাংশনকে অপ্টিমাইজ করব ডেটাসেট এর একেকটা রো নিয়ে।

আগে কোড দেখা যাক,

```

# Parameters to be updated
_theta_0 = 0.0
_theta_1 = 0.0
_theta_2 = 0.0

# Data Length
m = float(len(y))

# Epoch [No. of iterations]
epoch = 10

# Learning Rate
alpha = 0.01

# Costs
costs = []

# Initializing Stochastic Gradient Descent
for i in range(epoch):

    # Iterate over all of the data
    for j in range(len(y)):

        # Update theta_0 first
        _theta_0 = _theta_0 + (alpha / m) * (y[j] - predict(_theta_0, _theta_1, _theta_2, feature_1[j], feature_2[j]))

        # Use updated theta_0 to update theta_1
        _theta_1 = _theta_1 + (alpha / m) * (y[j] - predict(_theta_0, _theta_1, _theta_2, feature_1[j], feature_2[j])) * feature_1[j]

        # Again use theta_1 to update theta_2
        _theta_2 = _theta_2 + (alpha / m) * (y[j] - predict(_theta_0, _theta_1, _theta_2, feature_1[j], feature_2[j])) * feature_2[j]

    # Calculating cost
    cost = computeCost(_theta_0, _theta_1, _theta_2, feature_1, feature_2, y)

    # Saving it to the list for future use
    costs.append(cost)

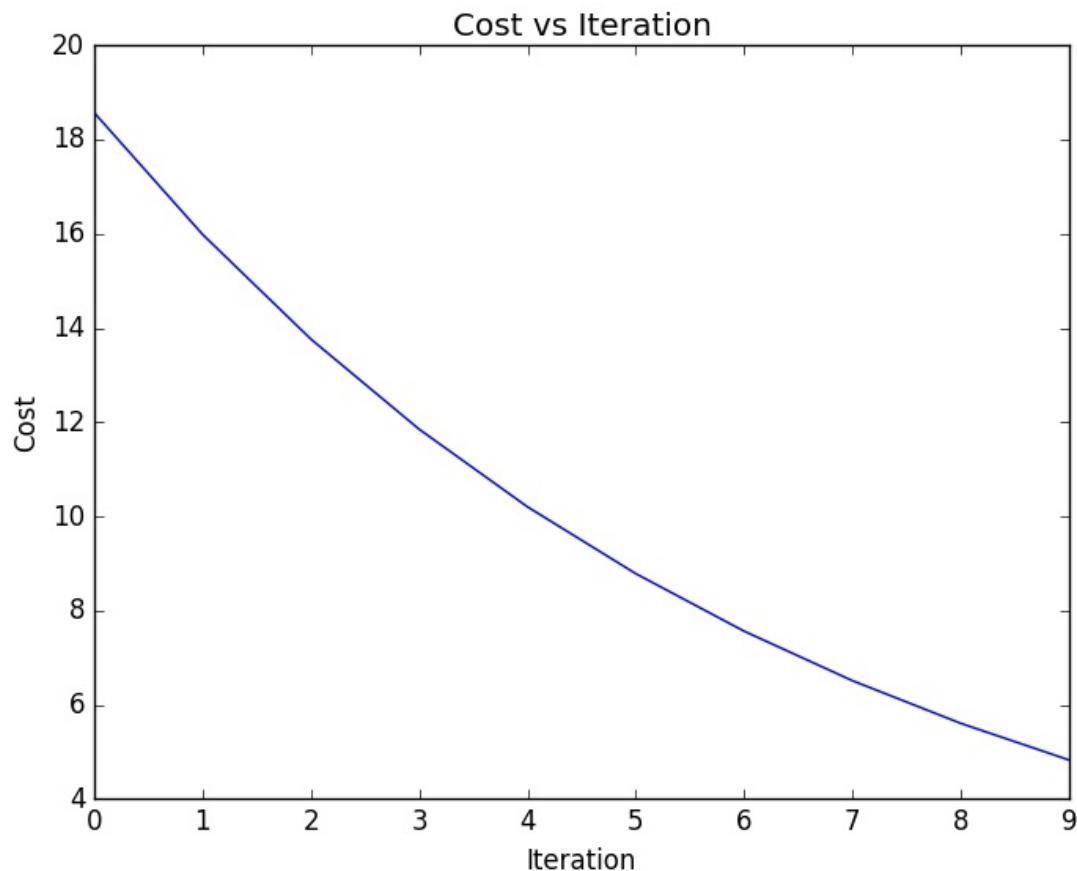
    # Printing cost after each epoch
    print("Cost: {}".format(cost))

# Plotting Cost vs Iteration Graph
plot_cost_vs_iteration(costs)

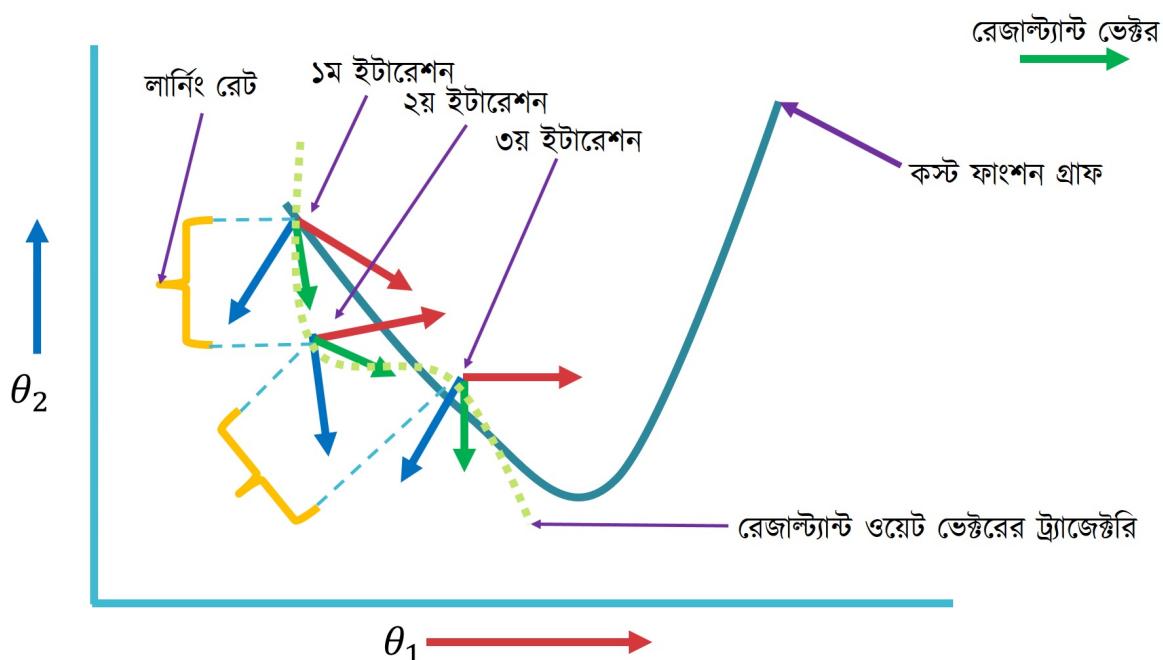
```

আমরা $epoch$ সম্পূর্ণ ডেটাসেট এ ব্যবহার না করে, প্রতি Row ধরে প্যারামিটার আপডেট করি।

কস্ট বনাম ইটারেশন গ্রাফ



স্টোক্যাস্টিক গ্রেডিয়েন্ট ডিসেন্ট কীভাবে কাজ করে?



মিনি ব্যাচ গ্রেডিয়েন্ট ডিসেন্ট

ব্যাচ গ্রেডিয়েন্ট ডিসেন্ট + স্টোক্যাস্টিক গ্রেডিয়েন্ট ডিসেন্ট = মিনি ব্যাচ গ্রেডিয়েন্ট ডিসেন্ট! এর মানে, ব্যাচ গ্রেডিয়েন্ট ডিসেন্ট ও স্টোক্যাস্টিক গ্রেডিয়েন্ট ডিসেন্ট উভয় অ্যালগরিদমের ভাল জিনিসগুলো নিয়ে এই অ্যালগরিদম ডেভেলপ করা হয়েছে। ওভারশট বা আভারশট সমস্যা ও কম্পিউটেশনাল কম্প্লেক্সিটির মধ্যে *Trade – off* করার জন্যই মূলত এর উৎপত্তি।

কোড ও আউটপুট দেখা যাক।

```

# Parameters to be updated
_theta_0 = 0.0
_theta_1 = 0.0
_theta_2 = 0.0

# Data Length
m = float(len(y))

# Epoch [No. of iterations]
epoch = 20

# Learning Rate
alpha = 0.01

# Costs
costs = []

# Mini Batch Gradient Descent
mini_batch_size = 5
mini_batches = int(m / mini_batch_size)

n = 0

for i in range(epoch):
    for batch in range(1, mini_batches + 1):
        n = batch * mini_batch_size
        for j in range(n):
            _theta_0 = _theta_0 + (alpha / m) * (y[j] - predict(_theta_0, _theta_1, _theta_2, feature_1[j], feature_2[j]))
            _theta_1 = _theta_1 + (alpha / m) * (y[j] - predict(_theta_0, _theta_1, _theta_2, feature_1[j], feature_2[j])) * feature_1[j]
            _theta_2 = _theta_2 + (alpha / m) * (y[j] - predict(_theta_0, _theta_1, _theta_2, feature_1[j], feature_2[j])) * feature_2[j]

        # Calculating cost
        cost = computeCost(_theta_0, _theta_1, _theta_2, feature_1, feature_2, y)

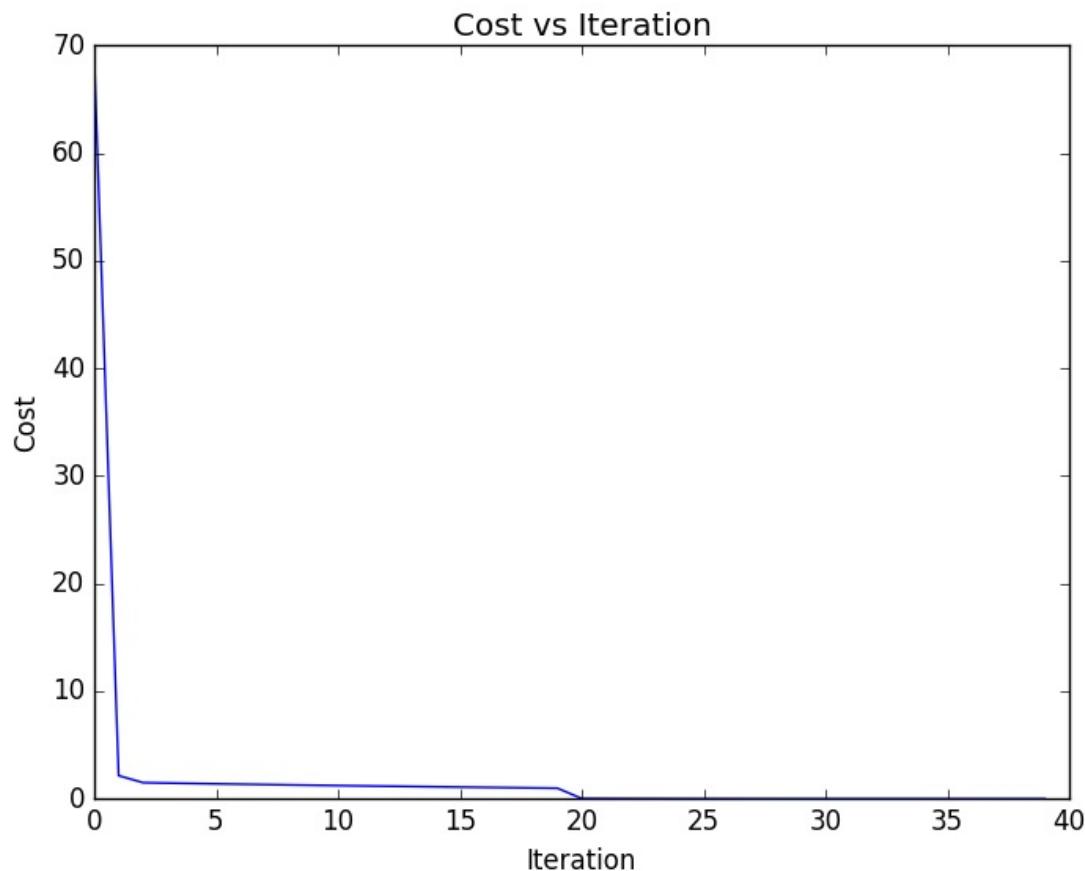
        # Saving it to the list for future use
        costs.append(cost)

    # Printing cost after each epoch
    print("Cost: {}".format(cost))

# Plotting Cost vs Iteration Graph
plot_cost_vs_iteration(costs)

```

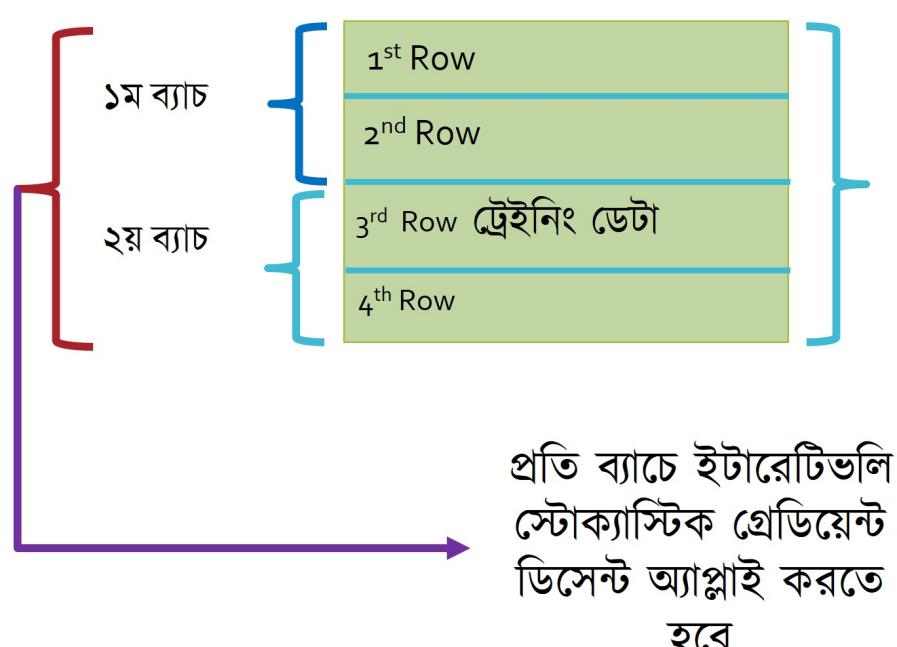
কষ্ট বনাম ইটারেশন গ্রাফ



মিনি ব্যাচে ডেটাসেট ভাগ করা

এখানে রো হচ্ছে ২০ টা, একে আমি ৫টি ব্যাচে ভাগ করেছি তাই প্রতি ব্যাচে রো সংখ্যা বা ডেটা সংখ্যা হল ৪।

অভাবতাল ধারণা নেয়ার জন্য নিচের ডায়াগ্রামটা দেখা যাক।



এই পর্ব এই পর্যন্তই। পরবর্তী পর্বে গ্রেডিয়েন্ট ডিসেটের নরমাল রিপ্রেজেস্টেশন দেখানো হবে।

প্র্যাক্টিক্যাল মাল্টিড্যারিয়েবল লিনিয়ার রিগ্রেশন : গ্রেডিয়েন্ট ডিসেন্টের নরমাল ফর্ম

গত চ্যাপ্টারে আলোচনা করা হয়েছিল গ্রেডিয়েন্ট ডিসেন্ট অপ্টিমাইজেশন অ্যালগরিদমের বিভিন্ন প্রকারভেদ নিয়ে। এই চ্যাপ্টারে আলোচনা করা হবে লিনিয়ার রিগ্রেশনের ক্ষেত্রে গ্রেডিয়েন্ট ডিসেন্টের নরমাল ফর্ম (Normal Form) নিয়ে। পশাপাশি নিউমেরিকাল সল্যুশন কেন নির্ভরশীল ও কী কী কারণে Normal Form ফেইলড হতে পারে সেটাও বিস্তারিত ব্যাখ্যা করা হবে।

বিগত কয়েকটা অধ্যায়ে লিনিয়ার রিগ্রেশনের প্যারামিটার আপডেটের ক্ষেত্রে নিউমেরিকাল অপ্টিমাইজেশনের সাহায্য নেয়া হয়েছিল। কিন্তু এমন কোন উপায় আছে যার মাধ্যমে ডেটাসেট বারবার ইটারেট না করে একবারই ক্যালকুলেশন করে প্যারামিটারের মান নির্ধারণ করা যায়? আমরা কি কোন চালাকি খাটিয়ে এর ম্যাথমেটিক্যাল এক্সপ্রেশন বের করতে পারি?

সেটা করার আগে দেখতে হবে আমাদের হাতে কী কী তথ্য আছে।

কস্ট ফাংশন

কস্ট ফাংশনের সূত্র দেখতে দেখতে নিচ্ছয়ই আপনারা বিরক্ত। কিন্তু বুঝতে হবে, মেশিন লার্নিংয়ের মূল উদ্দেশ্যই কিন্তু কস্ট ফাংশন ডিফাইন করে তার অপ্টিমাইজেশন করা। বিভিন্ন কাজের জন্য কস্ট ফাংশন বিভিন্ন রকম হয়, আর এটা গবেষণার একটা বিশাল অংশ জুড়ে আছে। কথা না বাড়িয়ে লেখা যাক,

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

ম্যাট্রিক্স আকারে কস্ট ফাংশন,

$$J(\theta) = \frac{1}{2m} \| X_{train} \theta - y_{train} \|^2$$

নোট:

ডাবল ডার্টিকেল বার দিয়ে Norm বুঝানো হয়েছে। যদি সাধারণত বলা না থাকে তাহলে ধরে নিতে হবে Norm টা L_2 লেভেল এর। অর্থাৎ, Euclidean Distance।

হাঈপোথিসিস ফাংশন

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

ম্যাট্রিক্স ফরম্যাটে,

$$h_{\theta}(X) = \theta^T X = X\theta$$

ট্রেইন ডেটার সাপেক্ষে,

$$h_{\theta}(X_{train}) = X_{train}\theta$$

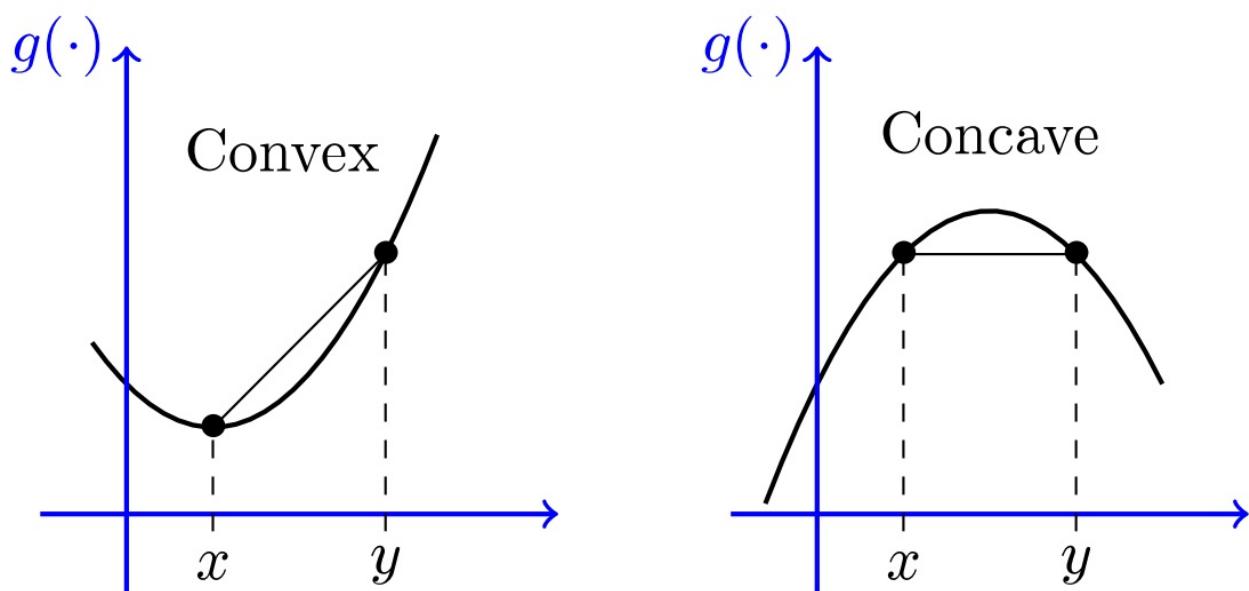
সুতৰাং ধৰি,

$$y = y_{train}$$

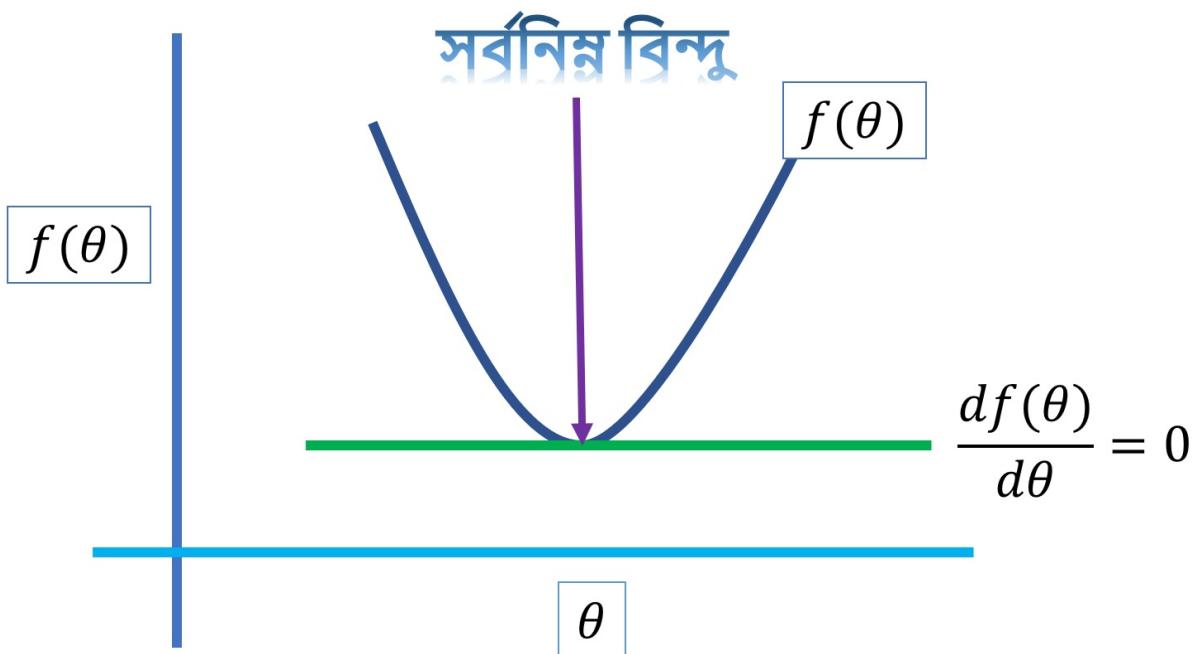
আরও কিছু গুরুত্বপূর্ণ তথ্য:

- কস্ট ফাংশন কনভেক্স আকারের তাই এর সবনিম্ন বিন্দু আছে

কনভেক্স ও কনকেভ ফাংশন দেখতে এরকম,



- $\theta_0, \theta_1, \theta_2$ এর একটা নির্দিষ্ট মান আছে যার জন্য কস্ট ফাংশনের মান শূন্য
- কোন ফাংশনের সবনিম্ন ও সর্বোচ্চ বিন্দুতে ঢাল শূন্য বা, ও বিন্দুতে ফাংশনের ডেরিভেটিভের মান শূন্য



এটাই কাজ আগামোর জন্য যথেষ্ট !

আমরা যেহেতু জানি কস্ট ফাংশনটি একটি কনভেক্স ফাংশন এবং এর একটি নিশ্চিত পয়েন্টে ডেরিভেটিভ শূন্য।
তাহলে এই ফাংশনের ডেরিভেটিভ সমান ০ ধরে আমরা যদি $\theta_0, \theta_1, \theta_2$ এর জন্য সমাধান করি তাহলেই আমরা
কাঞ্চিত প্যারামিটার ভ্যালু পেয়ে যাব। পুরো কাজটা করতে হবে ট্রেইনিং ডেটার উপরে তাই এখানে X_{train}
নোটেশন দিয়ে বোঝানো হচ্ছে ডেটাসেট এবং ট্রেইনিং ভাগ দিয়ে প্যারামিটারের মান বের করা হবে।

অর্থাৎ কিনা,

$$\nabla_{\theta} J(\theta) = 0 \\ \implies \nabla_{\theta} \left\{ \frac{1}{2m} \sum_{i=1}^m \left(y^{(i)} - h_{\theta}(x^{(i)}) \right)^2 \right\} = 0$$

উপরের থেকে ম্যাট্রিক্স নোটেশন ব্যবহার করে,

$$\implies \nabla_{\theta} \frac{1}{m} \|X_{train}\theta - y_{train}\|^2 = 0$$

$$\implies \frac{1}{m} \nabla_{\theta} \|X_{train}\theta - y_{train}\|^2 = 0$$

ম্যাট্রিক্সের সূত্রানুযায়ী, আমরা A^2 কে $A^T A$ আকারে লিখতে পারি। যেখানে A একটি যেকোন ডাইমেনশনের
ম্যাট্রিক্স ও A^T বলতে A ম্যাট্রিক্সের ট্রান্সপোজ বোঝায়। তাহলে উপরের কস্ট অংশ পরিবর্তন করে লেখা যায়,

$$\begin{aligned} &\implies \nabla_{\theta} (X_{train}\theta - y_{train})^T (X_{train}\theta - y_{train}) &= 0 \\ &\implies \nabla_{\theta} (\theta^T X_{train}^T - y_{train}^T) (X_{train}\theta - y_{train}) &= 0 \\ &\implies \nabla_{\theta} (\theta^T X_{train}^T X_{train}\theta - y_{train}^T X_{train}\theta - \theta^T X_{train}^T y_{train} - y_{train}^T y_{train}) &= 0 \end{aligned}$$

মাঝখানের টার্মের পার্শিয়াল ডেরিভেটিভ এরকম হবে ,

$$\nabla_{\theta} (y_{train}^T X_{train} \theta) = y_{train}^T X_{train}$$

$$\nabla_{\theta} (\theta^T X_{train}^T y_{train}) = X_{train}^T y_{train}$$

এবং এই ক্ষেত্রে,

```
>>> X = np.array([[1, 2, 3], [4, 5, 6]])
>>> y = np.array([1, 2])
>>> X.T.dot(y)
array([ 9, 12, 15])
>>> y.T.dot(X)
array([ 9, 12, 15])
```

$$y_{train}^T X_{train} = X_{train}^T y_{train}$$

সুতরাং,

$$\implies 2X_{train}^T X_{train} \theta - 2X_{train}^T y_{train} = 0$$

থিটার মান বের করতে হলে বাকি টার্ম ডানপাশে নিয়ে থিটার সাপেক্ষে এক্সপ্রেস করতে হবে,

$$\implies \theta = (X_{train}^T X_{train})^{-1} X_{train}^T y_{train}$$

তাহলে এটাই সেই গাণিতিক সূত্র, যাৰ মাধ্যমে ইটারেশন ছাড়াই কিছু ম্যাট্রিক্স অপারেশনের মাধ্যমে থিটার কাঞ্চিত মান পাওয়া যাবে। এবং যাকে সবাই **Normal Form of Gradient Descent** নামে চেনে!

এই সূত্র এবাৰ আমৰা [একটা ডেটাসেট](#) এৰ উপৰ অ্যাপ্লাই কৰব।

```

from __future__ import print_function
# Importing necessary libraries
import numpy as np

# Loading dataset
X = []
with open('ex3x.dat', 'r') as f:
    for line in f.readlines():
        f1, f2 = [float(row) for row in line.split()]

        # Adding a 1 to add a bias parameter, remember?
        X.append([1, f1, f2], axis=0)

num_data = len(X)
num_feature = 3

# Reshaping the input data matrix
X = X.reshape(47, 3)

y = np.array([])
with open('ex3y.dat', 'r') as f:
    for line in f.readlines():
        y.append(float(line))

# Reshaping output data matrix
Y = y.reshape(47, 1)

```

ডেটাসেট লোড করে রিশেপ করে নিলাম যাতে ম্যাট্রিক্স অপারেশন ঠিকঠাক করা যায়।

গ্রেডিয়েন্ট ডিস্ট্রেট নরমাল ফর্ম অ্যাপ্লাই করা

$$\theta = (X_{train}^T X_{train})^{-1} X_{train}^T y_{train}$$

```

# Getting parameter value
theta = np.linalg.inv(X.T.dot(X)).dot(X.T.dot(Y))

```

এই একলাইনের কোড রান করলেই থিটার মান পাওয়া যাবে। চমৎকার! তাই না?

Note: উপরের কোড না বোঝা গেলে "Numpy পরিচিতি" অধ্যায়টি পড়তে হবে।

প্রেডিকশন

থিটার মান তো পাওয়া গেল, এবার সেটা দিয়ে প্রেডিক্ষন করব কীভাবে? সহজ, হাইপোথিসিস ফাংশন অনুযায়ী ডেটাসেট থেকে যেকোন i^{th} ডেটা সিলেক্ট করে তার সাথে θ এর মান এর ডট প্রোডাক্ট নেব।

```
# If I want to predict for 1st data in the dataset we have then
y0 = X[0].dot(theta)
```

স্পেসিফিক ইনডেক্সের ডেটার প্রেডিকশনের জন্য কোড

```
idx = 0
print("Prediction for row {} in dataset : {}".format(idx + 1, X[idx].dot(theta)))
print("Real value for row {} in dataset: {}".format(idx + 1, Y[idx]))
print("Difference between actual value and prediction : {}".format(Y[idx] - X[idx].dot(theta)))

# Output:
# Prediction for row 1 in dataset : [ 356283.1103389]
# Real value for row 1 in dataset: [ 399900.]
# Difference between actual value and prediction : [ 43616.8896611]
```

যেহেতু আমরা বড় বড় মান প্রেডিষ্ট করছি তাই 43616.88966 ডিফারেন্স খুব একটা খারাপ প্রেডিকশন না। মনে রাখতে হবে, মান যত বড় হবে, ছোট ছোট পরিবর্তন আমরা ইগনোর করতে পারি।

প্রতিষ্ঠিত লাইব্রেরির সাথে এই প্রেডিকশনের তফাং কতটা?

এই ডেটাসেট ব্যবহার করে এখন দেখব সাইকিট-লার্নের লিনিয়ার মডেলের প্রেডিকশনের সাথে আমাদের তৈরি মডেলের প্রেডিকশনের তফাং কতটা।

```
# importing necessary library
from sklearn.linear_model import LinearRegression

# Select index
idx = 0

# Initializing the model
lr = LinearRegression()

# Fitting the model
lr.fit(X, Y)

# Predicting using sklearn
print("sklearn's prediction for row {} : {}".format(idx + 1, lr.predict(X[idx]).ravel()))
print("our model's prediction for row {} : {}".format(idx + 1, X[idx].dot(theta)))

# output
# sklearn's prediction for row 1 : [ 356283.1103389]
# our model's prediction for row 1 : [ 356283.1103389]
```

অসাধারণ! তারমানে আমরা এমন একটা অ্যালগরিদম তৈরি করলাম যেটা দিয়ে ডেটাসেট একবার ইটারেট করলেই আমরা থিটার মান পেয়ে যাব। গ্রেডিয়েন্ট ডিস্টের কোনই দরকার নাই। ম্যাথেমেটিশিয়ানরা কনডেক্স অপ্টিমাইজেশন নিয়ে এত চিত্তিত কেন?

তাই কি? আরেকটা উদাহরণ দেখা যাক।

পূর্বের অধ্যায়ের সিনথেটিক ডেটার উপরে এই ফরমুলা অ্যাপ্লাই করে

```
def make_fake_data(x1, x2):
    # y = 5+2*x1+3*x2
    return float(5 + 2 * x1 + 3 * x2)

# first feature of input data, col - 1
feature_1 = np.array([float(i) for i in range(1, 21)])
# Second feature of input data, col - 2
feature_2 = np.array([float(i) for i in range(21, 41)])

# Output
y = np.array([
    make_fake_data(f1, f2)
    for f1, f2 in
    zip(feature_1, feature_2)
])

# Making the input data matrix
x = np.array([np.ones(len(feature_1)), feature_1, feature_2]).T
```

এবার আগের মতই থিটা আপডেট করব এভাবে,

```
# Updating theta for our synthetic data
theta = np.linalg.inv(x.T.dot(x)).dot(x.T.dot(y))
```

সিনথেটিক ডেটার উপরে প্রেডিকশন

উপরের কোড রান করলেই থিটার মান পাওয়া যাবে, এবার আমরা এই থিটা দিয়ে টেস্ট করে দেখব প্রেডিকশন কেমন আসে!

```

idx = 0
print("Prediction for row {} in dataset : {}".format(idx + 1, x[idx].dot(theta)))

print("Real value for row {} in dataset: {}".format(idx + 1, y[idx]))

print("Difference between actual value and prediction : {}".format(y[idx] - x[idx].dot(theta)))

# Output
# Prediction for row 1 in dataset : 15.9375
# Real value for row 1 in dataset: 70.0
# Difference between actual value and prediction : 54.0625

```

অঙ্গুত! এত বাজে প্রেডিকশন! বাজেই বলা উচিং কারণ আমরা ট্রেইন ডেটার উপরেই প্রেডিকশন চালাচ্ছি, যদি ঠিকঠাক মডেল ফিট হয় তাহলে এর আরও কম আসার কথা। কিন্তু এখানে পুরো আকাশ পাতাল তফাহ!

সিনথেটিক ডেটার উপরে সাইকিট-লার্ন লাইব্রেরি দিয়ে প্রেডিকশন

```

# previous model is now fitted with our synthetic data
lr.fit(x, y)

idx = 0

# Predicting using sklearn
print("sklearn's prediction for row {} : {}".format(idx + 1, lr.predict(x[idx]).ravel()))

print("our model's prediction for row {} : {}".format(idx + 1, x[idx].dot(theta)))

print("Real output of row {} : {}".format(idx + 1, y[idx]))

# Output
# sklearn's prediction for row 1 : [ 70.]
# our model's prediction for row 1 : 15.9375
# Real output of row 1 : 70.0

```

তারমানে কিছু একটা গুগোল হয়েছে! আগেরবার কিন্তু প্রেডিকশন ঠিকঠাক ছিল কিন্তু এই ডেটাসেট এর উপরে প্রেডিকশন এত খারাপ কেন? এই কেন এর উত্তর জানতে হলে আরেকটু পড়তে হবে। সবসময় ফরমুলা ঠিকঠাক কাজ করে এমন কোন কারণ নেই।

মেশিন লার্নিংয়ে এধরণের সমস্যা ডিবাগ করার জন্য লিনিয়ার অ্যালজেব্রা, ক্যালকুলাস অহরহ ব্যবহার করা হয়। লার্জ স্কেল মডেল ডেপ্লয় করার আগে নিউমেরিকাল গ্রেডিয়েন্ট চেকিং অত্যাবশ্যিক।

আমাদের সল্যুশন কেন কাজ করছে না?

যিটা প্যারামিটারের সঠিক মানের জন্যই আমরা সঠিক প্রেডিকশন পাব। কিন্তু সঠিক প্রেডিকশন যেহেতু হয় নি তাই যিটা আপডেটও ঠিক ঠাক হয় নি।

যিটার সূত্র আবার লিখে পাই,

$$\theta = (X_{train}^T X_{train})^{-1} X_{train}^T y_{train}$$

এইখানের কোন একটা অপারেশনে সমস্যা হয়েছে। মনে রাখতে হবে, ম্যাট্রিক্স ইনভার্সনে প্রায়ই সমস্যা হতে পারে। সকল ম্যাট্রিক্স ইনভার্টিবল না।

$$(X_{train}^T X_{train})^{-1}$$

এই টার্মটা আগে অ্যানালাইজ করতে হবে। কোন ম্যাট্রিক্স ইনভার্টিবল কিনা সেটা দেখার জন্য আমরা তার ডিটার্মিনেন্ট (**Determinant**) নিয়ে থাকি।

ডিটার্মিনেন্ট এর মান যদি 0 হয় তাহলে বলব ম্যাট্রিক্সটি **Singular** এবং এটি ইনভার্টিবল না।

ইনভার্সনের ভিতরের টার্মের ডিটার্মিনেন্ট নিতে হবে, ম্যাথমেটিক্সের নোটেশনে,

$$\det(X_{train}^T X_{train})$$

প্রথমে লোডেড ডেটাসেট এর ডিটার্মিনেন্ট চেক করা যাক

```
# Checking the determinant of X
print(np.linalg.det(X.T.dot(X)))

# Output
# 24967305352.999886
```

এর মান অনেক বিশাল, তারমানে অবশ্যই প্রথম ডেটাসেট এর ট্রান্সপোজ ও নিজের সাথের ডট প্রোডাক্ট ইনভার্টিবল।

সিনথেটিক ডেটাসেট এর ডিটার্মিনেন্ট

পরেরটা চেক করা যাক,

```
# Checking the determinant of x
print(np.linalg.det(x.T.dot(x)))

# Output
# 6.6151528699265518e-09
```

হ্ম, এখানে কিন্তু ডিটার্মিনেন্ট এর মান প্রায় 0 এর কাছাকাছি, যেহেতু পুরোপুরি শূন্য না তাই একে ইনভার্ট করা গেছে। একে আমরা শূন্যই ধরতে পারি, আর সেটা চিন্তা করলে পরবর্তীতে যে সল্যুশন পাওয়া গেছে সেটা আমরা চাই নি।

এই কারণেই মূলত নিউমেরিকাল অপ্টিমাইজেশন সবসময় নির্ভরযোগ্য, আশাৰ কথা হল এই ফৰমুলা বাস্তৱ ডেটাসেট এ ভালভাবেই কাজ কৰবে, কেননা; বাস্তৱ ডেটাসেট এ প্রতিটা অবজারভেশন কখনো এভাৱে জেনাৱেটেড হবে না, তাতে প্ৰচুৰ নয়েজ থাকবে তাই তাদেৰ ডিটামৰ্সিনেন্ট ০ আসাৰ সম্ভাবনা কম!

নোট

সিস্টেম অফ লিনিয়ার ইকুয়েশনস ও সিঙ্গুলার ম্যাট্রিক্স (Singular Matrix)

ম্যাট্রিক্স হল ডেষ্টের কালেকশন। যখন ম্যাট্রিক্সের সকল ডেষ্টের সবগুলোৱ অবস্থান ঠিক একই জায়গায় হয় সেক্ষেত্ৰে আমৰা বলব, সেই ম্যাট্রিক্স দ্বাৰা গঠিত লিনিয়ার সিস্টেমেৰ সল্যুশন অসীম সংখ্যক।

মানে,

$$\begin{aligned}x + y + z &= 5 \\2x + 2y + 2z &= 10 \\3x + 3y + 3z &= 15\end{aligned}$$

আপনাকে যদি বলা হয়, এই ইকুয়েশন তিনটা সল্ভ কৰুন, তাহলে আপনি কি পাৰবেন? লক্ষ্য কৰে দেখবেন এখানে আসলে ইকুয়েশন একটাই, সেক্ষেত্ৰে একটা ইকুয়েশন দিয়ে তিনটা ভ্যারিয়েবলেৰ মান কীভাবে বেৱ কৰা যায়?

এখানে, x, y, z এৰ বিভিন্ন মানেৰ জন্য আপনি ৫ পাৰবেন, আমি যদি বলি, $x = 1, y = 2, z = 2$ আপনি বলতে পাৰবেন, $x = 2, y = 1, z = 2$!

তিনটা ডেষ্টেৰ যেহেতু একই স্থানে অবস্থান কৰছে (গুধু একটাৰ স্কেলিং ফ্যাক্টৰ আৰেকটা থেকে বেশি) তাই ডেষ্টেৰ তিনটাৰ উপৱেৰ সবগুলো বিন্দুই এৰ এক একটি সল্যুশন। আৱ এৱকম হলে আমৰা সিঙ্গুলার ম্যাট্রিক্স বলি, যাৱ ডিটামৰ্সিনেন্ট হয় শূন্য।

উপৱেৰ ইকুয়েশনকে আমৰা ম্যাট্রিক্স আকাৰে প্ৰকাশ কৰে তাৰ ডিটামৰ্সিনেন্ট নিয়ে দেখি,

```
test_matrix = np.array([[1, 1, 1], [2, 2, 2], [3, 3, 3]])
print(np.linalg.det(test_matrix))

# Output
# 0.0
```

আপনি যদি পুৰোটা অধ্যায় পড়ে ও বুঝে থাকেন তাহলে বলব, নিউমেরিকাল অপ্টিমাইজেশন কেন গুৰুত্বপূৰ্ণ এবং কীভাবে একটা মেশিন লাৰ্নিং অ্যালগৱিদম ডিবাগ কৰা যায় সেটাৰ সম্পর্কে কিছুটা জ্ঞান লাভ কৰেছেন।

পরবর্তী অধ্যায়ে লজিস্টিক রিপ্রেশনের জন্য প্রযোজনীয় ইন্টুইশন বিন্দ করব।

লজিস্টিক রিপ্রেশন : পরিচিতি

বেশ কয়েকটি অধ্যায়ে আমরা কোন কিছুর মান প্রেডিট করার চেষ্টা করেছি লিনিয়ার মডেল ব্যবহার করে, যাকে লিনিয়ার রিপ্রেশন বলা হচ্ছিল। এখন থেকে পরবর্তী বেশ কিছু অধ্যায়ে আলোচনা করা হবে লজিস্টিক রিপ্রেশন, বা ক্লাসিফিকেশন সমস্যা নিয়ে। ক্লাসিফিকেশন সমস্যা রিপ্রেশন সমস্যার থেকে একটু ডিম্ব এবং বেশ কিছু ক্ষেত্রে জটিল। শুরুর দিকে সহজ সরল মডেল নিয়ে আলোচনা করার পর শেষের অধ্যায়গুলিতে মাল্টিক্লাস রিপ্রেশন বা সফটম্যাক্স রিপ্রেশন, লিনিয়ার ডিস্ক্রিমিনেট অ্যানালাইসিস এর ম্যাথমেটিক্যাল ফাউল্ডেশন নিয়ে বিস্তারিত আলোচনা করা হবে।

সাধারণত মেশিন লার্নিং সমস্যা দুই ধরণের হয়, মান প্রেডিট করা অথবা ক্লাসিফাই করা। নিম্নের সমস্যাগুলো ক্লাসিফিকেশন সমস্যার অঙ্গর্গত।

ক্লাসিফিকেশন সমস্যা

- ইমেইল : স্প্যাম নাকি সাধারণ?
- অনলাইন ট্র্যান্যাকশন : ফ্রড (হ্যাঁ বা না)?
- টিউমার : বিনাইন না ম্যালিগন্যাট (ক্যান্সার)?
- সেন্টিমেন্ট : পজিটিভ কথা নাকি নেগেটিভ?

অর্থাৎ এতদিন আমাদের উত্তর এসেছিল একটা মান হিসেবে কিন্তু এখানে আসবে লজিকাল মান, যেমন 1, 0।

ইন্টেলিজেন্ট রেস্টোরাঁ রিভিউ সিস্টেম

ধরুন, আপনি মেশিন লার্নিং প্র্যাক্টিশনার এবং আপনার বন্ধু বিশাল টাকাওয়ালা ব্যক্তি। সে একটি ওয়েবসাইট বানাতে চায় যেখানে বিভিন্ন রেস্টুরেন্টের রিভিউ ও রেটিং থাকবে। প্রতিটা রিভিউ পড়ে যাতে একজন ভিসিটরের গণণা করা না লাগে যে কয়টি নেগেটিভ ও পজিটিভ রিভিউ আছে এবং একটা রেস্টুরেন্ট সম্পর্কে সহজেই ওভারআল ধারণা পেয়ে যায় তাই আপনার বন্ধু আপনার কাছে আস্দার করল একটি সিস্টেম বিন্দ করে দিতে যেটা কিনা অটোমেটিক রিভিউ গুলো থেকে পজিটিভ ও নেগেটিভ রিভিউ আলাদা করবে এবং একটি কাউন্টারে শো করবে।

কয়েকটা উদাহরণ দেখা যাক,



Bikash3150
Dhaka City, Bangladesh
Level 4 Contributor

40 reviews
 18 restaurant reviews

"Star of Dhaka"

Reviewed 4 weeks ago

Star is one of the best biryani serving in dhaka from very long time. It is renown for its biryani. Also serves many sort of Bengali foods.

Helpful?

Thank Bikash3150

Report



Asaduzzaman R
Level 2 Contributor

5 reviews
 4 restaurant reviews

"Perfect place for foodies at very reasonable cost!"

Reviewed April 11, 2017 via mobile

The restaurant is famous for its dishes at very affordable price. Large group can easily accommodate. In this case reservation is required other seats will not be available. One shirt tasty food at cheaper price.

Helpful?

Thank Asaduzzaman R

Report



Luke S
Dhaka
Level 4 Contributor

24 reviews
 7 restaurant reviews
 5 helpful votes

"Food is tasty but service is not good."

Reviewed April 9, 2017

The food of Star Kabab and Restaurant is tasty but service is not good. Service staff takes too much time to serve food. Also the restaurant outlets are much crowded and there is no kids' zone, not suitable for families with children.

Helpful?

Thank Luke S

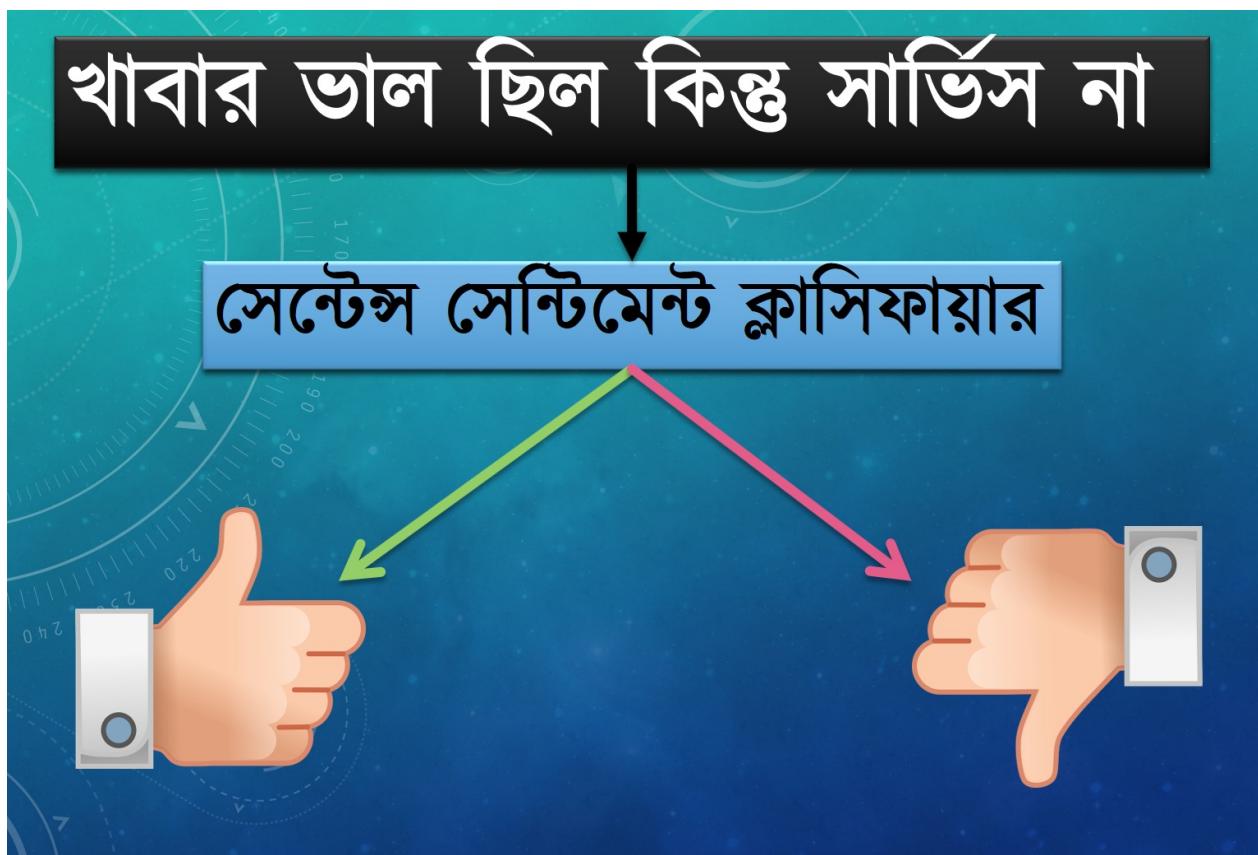
Report

রিভিউগুলো স্টার কাবাবের এবং সংগ্রহ করা হয়েছে ট্রিপ অ্যাডভাইজর ওয়েবসাইট থেকে

আপনার কাজ হবে এমন একটি মডেল তৈরি করা যে এই রিভিউ গুলোকে পজিটিভ ও নেগেটিভ হিসেবে স্লাসিফাই করতে পারে।

[অবশ্যই বেটিং থেকে আলাদা করা যায় কিন্তু আমরা এখানে রিভিউ থেকে স্লাসিফাই করার চেষ্টা করব]

সিস্টেম ওভারভিউ



লিনিয়ার ক্লাসিফায়ার : ইনটুইশন

এই সমস্যা কীভাবে সল্ভ করা যায়? আমরা একটা কাজ করতে পারি, একটা Weight টেবিল তৈরি করে তাতে ভাল ভাল শব্দ যেমন ভান , খ্রাঁধারণ , অনন্ধ্রাধারণ শব্দগুলোকে তাদের পজিটিভিটি অনুযায়ী পজিটিভ মান দিয়ে দিতে পারি এবং , খাবাপ , বাজে , না ইত্যাদি শব্দগুলোকে নেগেটিভ মান দিতে পারি।

এভাবে একটি টেবিল তৈরি করা যায়।

শব্দ	Weight
ভাল	1.0
অসাধারণ	2.0
অনন্যসাধারণ	2.5
খাবাপ	-1.0
বাজে	-1.1
জঘন্য	-2.0
না	-1.5
এটা, ওটা, আমি, কখন, কোথায়	0.0

Weight এর ভিত্তিতে বাক্য কে স্কোরিং করা

এবাব আমরা যদি বাক্যের এই `Weight` এর ভিত্তিতে উপর্যুক্ত বাক্যটির সেন্টিমেন্ট ল্লাসিফাই করতে চাই তাহলে সেটা কীভাবে করব?

সহজ, একটা ডিকশনারিতে আমরা শব্দকে `Key` এবং তার `Weight` কে ড্যালু হিসেবে রাখব এবং একটা লুপ চালিয়ে প্রতিটা শব্দের বিপরীতে যে `Weight` আছে সেগুলো যোগ করতে থাকব। যেটা হবে এই বাক্যের স্কোর `Score`।

কাজটা করার জন্য একটা ছোট পাইথন স্ক্রিপ্ট লেখা যাক,

```
# Simple Linear Classifier
words = [u'ভাল', u'জ্যার্বারণ', u'জনস্থার্যারণ', u'ধারাগ', u'বার্জ', u'জ্যন্ত্র', u'না', u'কিন্তু']
weights = [1.0, 2.0, 2.5, -1.0, -1.1, -2.0, -1.5, 0.0]

# Building vocabulary given the list of words and weights
vocabulary = {word : weight for word, weight in zip(words, weights)}

sentence = u'ধারার ভাল হিল কিন্তু গ্রাহিত না'

# If a word is not present in the vocabulary we simply ignore it
score = 0.0
for word in sentence.split(' '):
    if word in words:
        score = score + vocabulary[word]
    else:
        pass

print("Score: {}".format(score))

if (score > 0) : print("Sentence is positive")
else: print("Sentence is negative")
```

এটার আউটপুট আসলো,

```
Score: -0.5
Sentence is negative
```

এখনে আমি যেটা করলাম সেটা হল, স্কোর যদি সামগ্রিকভাবে পজিটিভ হয় তাহলে আমি ধরে নিচ্ছি বাক্যটি পজিটিভ সেন্টিমেন্টের মধ্যে পড়ে। যদি স্কোর ঋণাত্মক হয় তাহলে বুঝব বাক্যটি নেগেটিভ! এইয়ে আমি `Score > 0` হলে একটা ডিসিশন এবং `Score < 0` হলে আরেকটি ডিসিশন নিচ্ছি বা বাট্টারি সেট করছি এটার আরেকনাম (**স্প্যালার অ্যালার্ট**) ডিসিশন বাট্টারি। একটা ডিসিশন কখন রিজেক্ষন হবে, রিজেকশন রিজিওন কোনটা, কেন, কীভাবে সেটা নিয়ে পরে আলোচনা করা হবে।

সমস্যা

এই পদ্ধতিতে বেশ কিছু সমস্যা আছে, প্রথমত: বাংলাভাষার শব্দভাগের প্রচুর শব্দ আছে যেগুলো কিনা পজিটিভ, নেগেটিভ বা নিউট্রাল। এতশত শব্দ বের করা, তাদের তীব্রতা হিসেবে weight অ্যাসাইন করাও খুব জটিল, দীর্ঘমেয়াদী ও একধর্ম্মে কাজ। শুধু তাই নয়, আরেকটি বিশাল সমস্যা আছে যেটা একটু পরেই দেখানো হবে।

লিনিয়ার লাসিফায়ারের জ্যামিতিক ইন্টারপ্রিটেশন

আমরা যে সেটেজ লাসিফায়ার তৈরি করলাম, যদিও এটা বেশ সাধারণ এবং ব্যবহার উপযোগী নয় তাও এর একটা সুন্দর জ্যামিতিক ইন্টারপ্রিটেশন আছে যেটা বোঝা খুবই দরকারী। এটা বোঝার জন্য আমরা চলে যাব আমাদের HSC এর জ্যামিতি বইয়ের একটি অধ্যায়ে, যার নাম 'সরলরেখা'। সরলরেখা অধ্যায়ে বেশ কিছু জিনিস পড়ানো হয় কিন্তু বিশাল আফসোসের বিষয় হচ্ছে এই টপিক গুলো কোথায় অ্যাপ্লাই করা হয়, আদৌ দরকারী কিনা সেটা বলা হয় না। তেমনি একটি উপ-অধ্যায় হল একটি বিন্দু সরলরেখার কোন দিকে অবস্থান করছে। যদি না বোঝা যায় তাহলে নিচের চিত্র থেকে ব্যাখ্যা করা যাক।

একটা সরলরেখার সমীকরণ চিন্তা করা যাক, $y = m \times x + c$ । আরও মনে করি, $m = 2, c = 3$

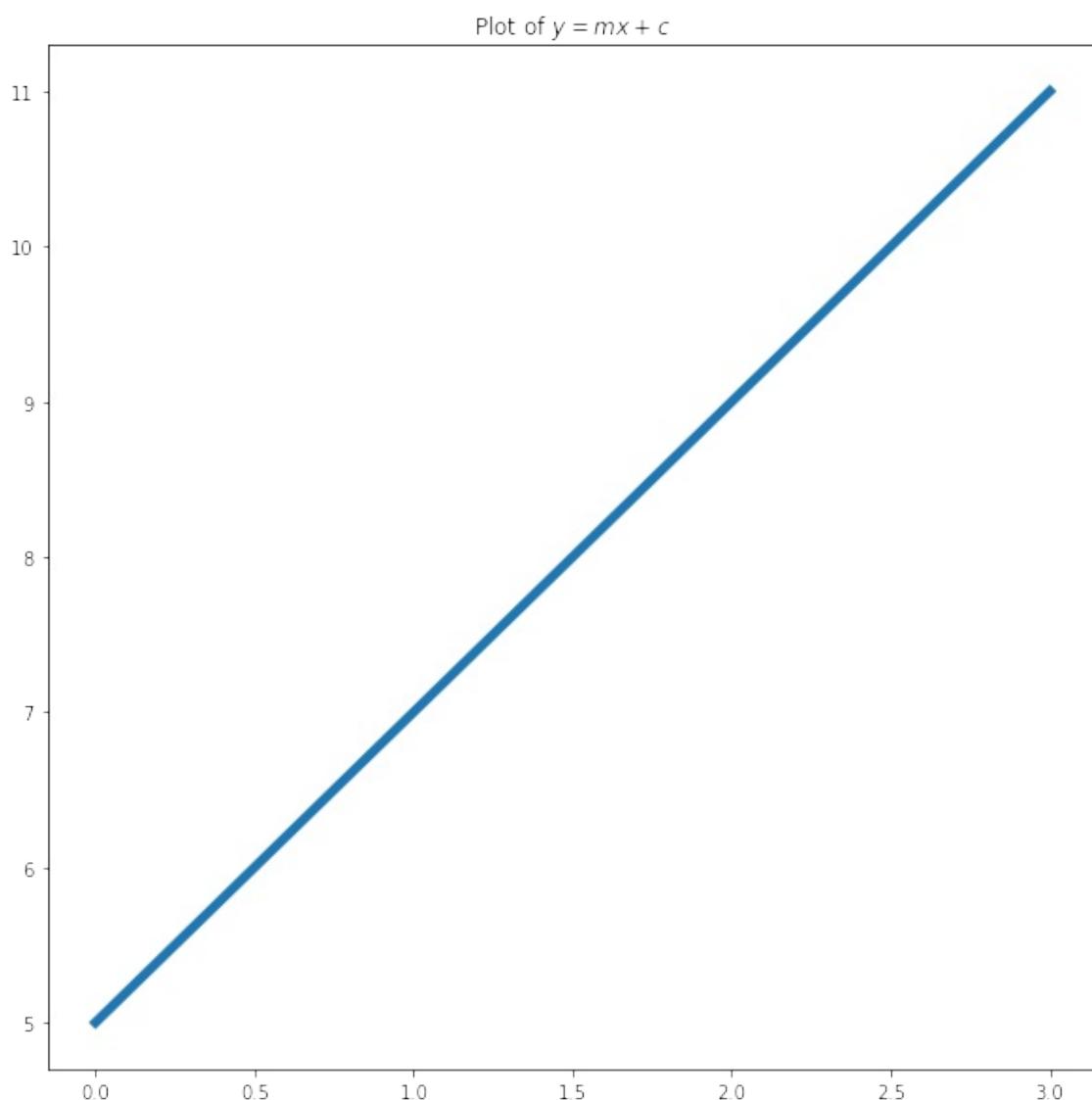
তাহলে আমি যদি x এর চারটি মানের কথা চিন্তা করি, $x = \{1, 2, 3, 4\}$ তাহলে y এর মানগুলি কত হবে?

```
m = 2
x = [1, 2, 3, 4]
c = 3
y = [(m*i + c) for i in x]

print(y)

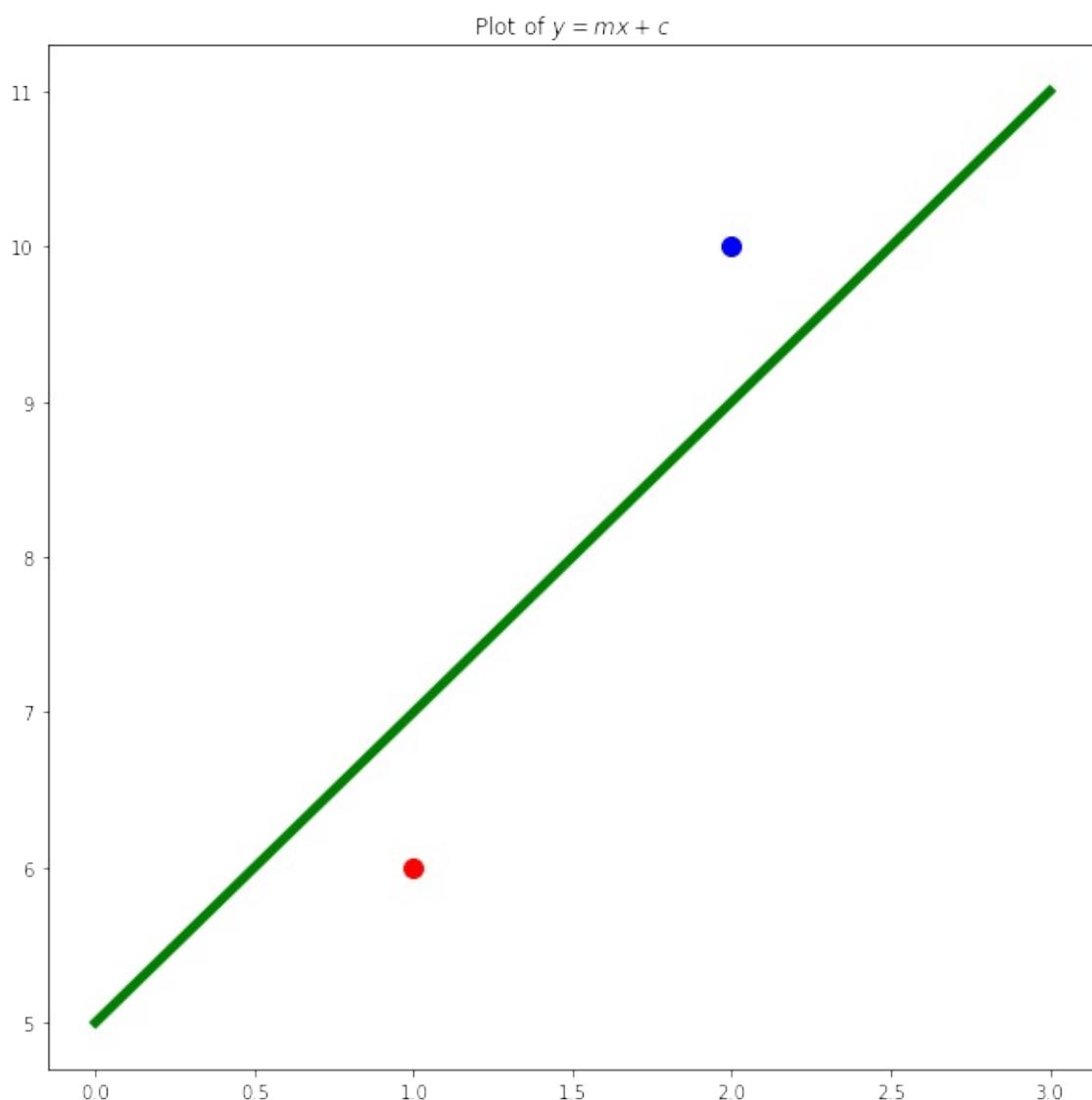
# Output
## [5, 7, 9, 11]
```

একে প্লট করা হলে,



এবাব এই গ্রাফে দুইটা বিন্দু প্লট করি, যদের কোঅর্ডিনেট যথাক্রমে, $(x_1, y_1) = (1, 6)$ এবং $(x_2, y_2) = (2, 10)$

```
plt.plot(y, linewidth="5", color='g')
plt.scatter(1, 6, color='r', s=100)
plt.scatter(2, 10, color='b', s=100)
plt.title('Plot of ' + r'$y=mx+c$')
```



দৃশ্যত, লাল বিন্দুটি সরলৰেখাটিৰ নিচে এবং নীল বিন্দুটি সরলৰেখার উপরে। কিন্তু গাণিতিকভাৱে কীভাৱে আপনি সম্পর্ক স্থাপন কৰতে পাৰবেন?

এইচএসসি এৰ জ্যামিতি না মনে থাকলেও সমস্যা নাই। নিচেৰ সূত্ৰটা দিয়ে সহজেই বেৱ কৰা যাবে,

$$\begin{aligned}y - mx - c &> 0 \\y - mx - c &< 0\end{aligned}$$

(x_1, y_1) এৰ জন্য ক্যালকুলেট কৰা যাক,

$$\begin{aligned}y_1 - mx_1 - c &= 6 - 2 \times 1 - 3 \\&= -2 < 0\end{aligned}$$

এবং (x_2, y_2) এৰ জন্য ক্যালকুলেট কৰলে,

$$\begin{aligned}y_2 - mx_2 - c &= 10 - 2 \times 2 - 3 \\&= 3 > 0\end{aligned}$$

সংক্ষেপে আমার যদি ইনপুট (x) এবং আউটপুট (y) জানা থাকে তাহলে আমরা এমন একটি সরলরেখার সমীকরণ বের করতে পারব (যদি ডেটা একটা সরলরেখা দিয়ে আলাদা করা যায়) যেটা কিনা এই ডেটাসেট ক্লাসিফাই করতে পারে। x, y দেয়া থাকলে কাজ হল m, c এর মান বের করা।

এই খিওরি যে শুধু সরলরেখার ক্ষেত্রে সত্য তাই নয়, ডেটার স্পেস n ডিমেনশনাল হলে তাকে $n - 1$ ডিমেনশনাল হাইপারপ্লেন দিয়ে আলাদা করা যাবে। (সবসময় না, ডেটাসেটের উপর নির্ভর করে) যেমন $3D$ এর ক্ষেত্রে ডেটাসেটের (x_1, x_2, y) এই তিনটি কোঅর্ডিনেট থাকবে যেখানে দুইটি হল ফিচার ডেটার আরেকটি হল আউটপুট এবং একে সেপারেট করার জন্য $3 - 1$ বা 2 ডিমেনশন বিশিষ্ট লাইন লাগবে যাকে কিনা বলা হয় **Plane**।

কার্নেল বেজড মেথডগুলোতে এই খিওরি অহরহ ব্যবহার করা হয়। (পাশাপাশি আরও অনেক খিওরি) যেটা নিয়ে আবারও পরে একসময় আলোচনা করা হবে।

অনেক কথাবার্তা হল, এবার আসল পয়েন্টে আসা যাক। আমরা কি লিনিয়ার রিগ্রেশন ব্যবহার করে ক্লাসিফাই করতে পারি না? কেন, ওখানেও তো আমরা এমন একটা সমীকরণ বের করি $y = \theta_0 + \theta_1 x$, তাহলে আমরা Mean Square Error ক্যালকুলেট করে সহজেই তো প্যারামিটার আপডেট করে এটা দিয়েই ক্লাসিফাই করতে পারি! তবে কেন লজিস্টিক রিগ্রেশন?

লিনিয়ার রিগ্রেশন দিয়ে ক্লাসিফাই করা যাবে কি?

(স্পষ্টলার অ্যালার্ট) : না!

কেন?

আরেকটি হাইপোথেটিক্যাল ডেটাসেট এর কথা চিন্তা করা যাক। টিউমারের আকারের ডিত্তিতে সেটা ম্যালিগন্যান্ট (খুবই ক্ষতিকর) কিনা সেটাৰ ডেটাসেট। এই কোডে আমি ডেটাসেট তৈরি করে লাইব্রেরি ব্যবহার করে লিনিয়ার রিগ্রেশন চালাব। যেহেতু আমরা বেসিক জানি তাই লাইব্রেরি ব্যবহার করতে সমস্যা নেই।

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(color_codes=True)

X_class_1 = np.arange(0, 50, 5)
X_class_2 = np.arange(70, 119, 5)
X_outlier = np.array([150])

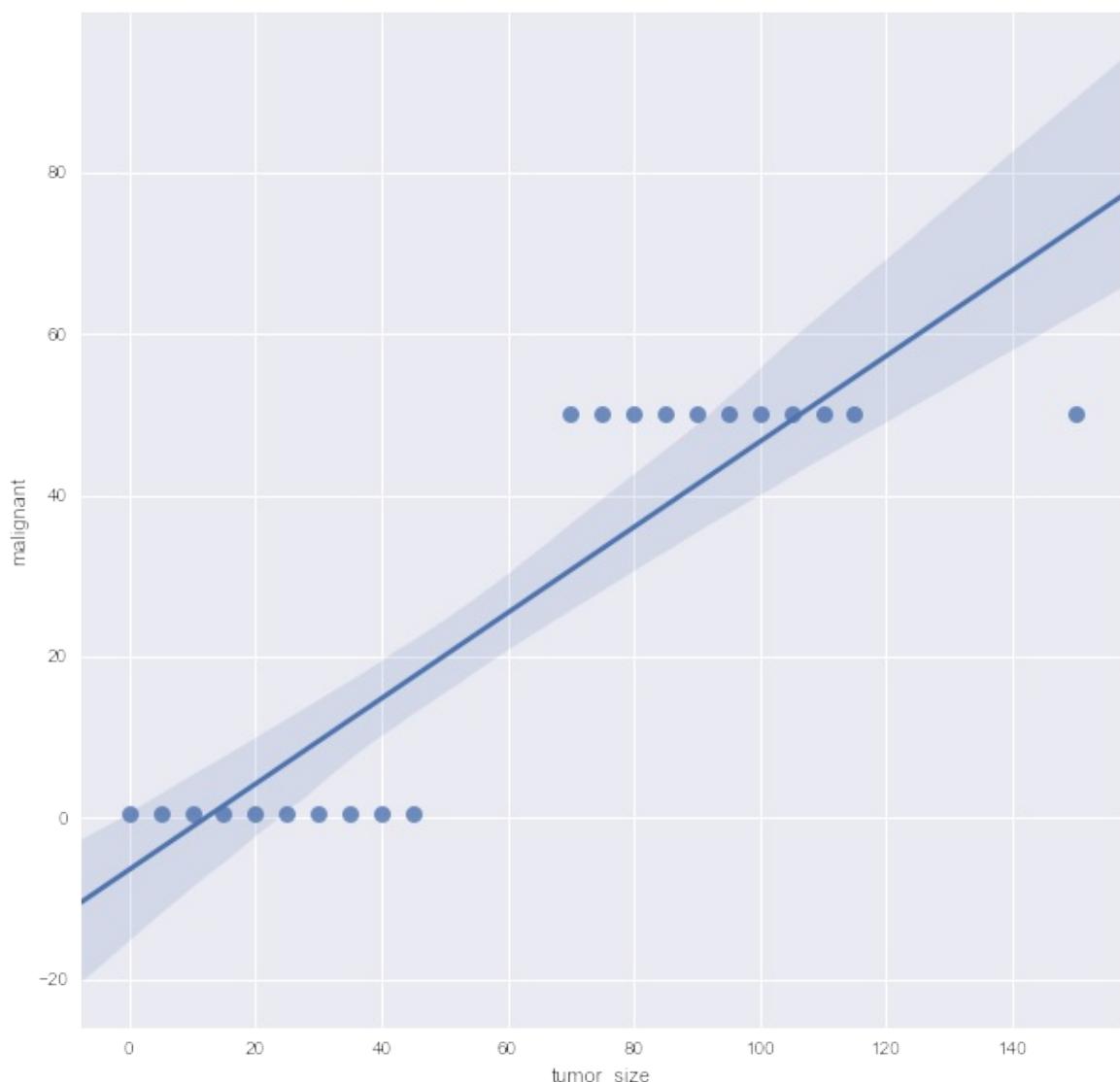
X = np.concatenate((X_class_1, X_class_2, X_outlier))

# Create Y data set
Y_class_1 = np.array([.5] * len(X_class_1))
Y_class_2 = np.array([50.0] * (len(X_class_2) + 1))
Y = np.concatenate((Y_class_1, Y_class_2))

synthetic_data = {
    "tumor_size" : pd.Series(X),
    "malignant" : pd.Series(Y)
}

df = pd.DataFrame(synthetic_data)
sns.regplot(x="tumor_size", y="malignant", data=df, scatter_kws={"s": 80})
```

কোডে প্রথমে ক্লাস-১ এর জন্য আমি কিছু ফিচার জেনারেট করলাম *np.arange* দিয়ে, এবং ক্লাস-২ এর জন্যও তাই। এবং আমি আরেকটা ডেটা তৈরি করলাম যেটার নাম দিলাম *outlier*। *outlier* বলার কারণ হল, এর অবস্থান আমি বেশ একটু দূরেই দিলাম। এবং এর কারণে লিনিয়ার রিপ্রেশন চালালে আউটপুট আসবে এরকম!



সমস্যাটা বোঝা যাচ্ছে? এই রিপ্রেশন লাইনে হ্যাত লস মিনিমাম কিন্তু মিসক্লাসিফিকেশন রেট অনেক অনেক বেশি।
আর এটা হওয়ার আরেকটা কারণ **outlier**। এই ডেটার কারণে লাইনটা এমনভাবে ফিট করেছে যে একই ক্লাসের
মধ্যে সে ভাগ করে ফেলছে!

প্রতিকার?

এই সমস্যার সমাধান করা খুব সহজ! আউটপুটকে কোন একটা মান না দিয়ে আমরা যদি তাৰ ৰেঞ্জ $[0, 1]$ এর মধ্যে
নিয়ে আসি? তাহলে নিশ্চয়ই সমস্যা হবে না? এমন একটা ফাংশন তৈরি করতে হবে যেটা নেগেটিভ মানকে শূন্যের
কাছাকাছি নিয়ে আসে এবং পজিটিভ মানকে ১ এর কাছাকাছি নিয়ে যায়। তাহলে ০ আউটপুট আসলে আমরা
বলতে পারব এই ডেটা এক ক্লাসের অঙ্গর্গত এবং ১ আউটপুট আসলে বলতে পারব সেটা আরেক ক্লাসের।

এইরকম অনেক ফাংশন আছে, তবে সাধারণত এই ফাংশনটা ব্যবহার করা হয়, যার নাম হল **logit** ফাংশন বা
sigmoid ফাংশন, এর ডোমেইন $(-\infty, \infty)$ এবং ৰেঞ্জ $(0, 1)$ ।

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

এৰ প্ৰোপার্টি প্লট কৰেও দেখা যায়,

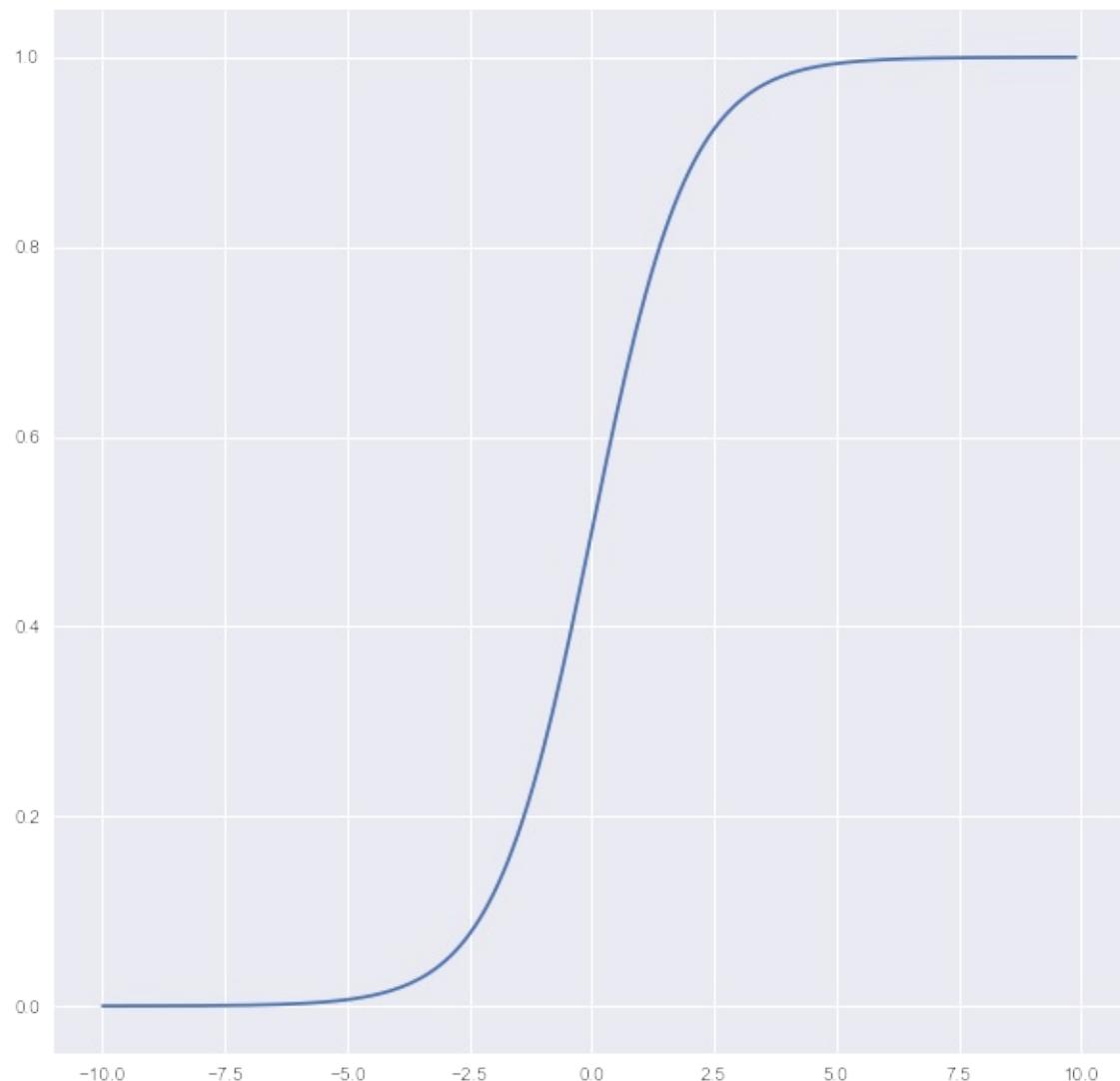
```
import numpy as np

def sigmoid(z):
    return 1.0 / ( 1 + np.exp(-z))

test_sig = np.arange(-10, 10, .1)
sig_out = sigmoid(test_sig)

plt.plot(test_sig, sig_out)
plt.show()
```

আউটপুট



তাহলে আউটপুট এটা করে লিনিয়ার রিপ্রেশন চালালেই তো এবার হবে তাই না? আবারও স্পষ্টভাবে অ্যালার্ট! না! আমাদের আবও একটু ক্রিয়েটিভ হতে হবে লজিস্টিক রিপ্রেশন কাজ করানোর জন্য। পাশাপাশি প্রযোবিলিটি, স্ট্যাটিস্টিক্স ও ইনফর্মেশন থিওরির বিল্ডিং ব্রকগুলো সম্পর্কে ধারণা রাখতে হবে। যা নিয়ে বিস্তারিত পরবর্তী অধ্যায়ে আলোচনা করা হবে।

লজিস্টিক রিপ্রেশন : প্রব্যাবিলিটি, বানুলি ও বাইনোমিয়াল ডিস্ট্রিবিউশন

গত অধ্যায়ে আমরা দেখেছিলাম, ক্লাসিফিকেশন প্রবলেমগুলো লিনিয়ার রিপ্রেশন দিয়ে করলে কী কী সমস্যা হতে পারে। পাশাপাশি আলোচনা করা হয়েছিল ডিসিশন বাট্টারি, লগিট বা সিগময়েড ফাংশন নিয়ে।

এই ব্যাপারে সবার একমত হওয়া উচিত যে, ক্লাসিফিকেশন সমস্যা সল্ভ করতে হলে একটু ডিম উপায়ে আগাতে হবে। কারণ, আমরা যে লস ফাংশন নিয়ে কাজ করছি, সেটা মিসক্লাসিফিকেশন মাপার জন্য ভাল না।

প্রশ্ন হচ্ছে আমরা তাহলে ক্লাসিফিকেশন এর টা মেজার করব কীভাবে? সেটা নিয়েই আজকের অধ্যায়।

আজকের টপিক:

- প্রব্যাবিলিটি ও ক্ষেত্রাল প্রব্যাবিলিটি
 - ইনডিপেন্ডেন্স
 - প্রব্যাবিলিটির সাম ও প্রোডাক্ট রুল
 - Bernoulli ট্রায়াল
 - Bernoulli ডিস্ট্রিবিউশন
 - Binomial ডিস্ট্রিবিউশন
-

প্রব্যাবিলিটি

ফরমাল ইক্যুয়েশন নিয়ে কথা না বলে বরং একটা উদাহরণ দিয়ে ব্যাখ্যা করা যাক। আমি এখানে গ্রাফিক্যাল মডেল দিয়ে ব্যাখ্যা করার চেষ্টা করব।

খুনের রহস্য

ধরি একটা বাড়িতে বাড়ির মালিক খুন হল। আমাদের প্রব্যাবিলিটিক অ্যাপ্রক্রিমেশন দিয়ে বের করতে হবে সেটা কে করেছে।

আরও ধরা যাক সাসপেন্ট দুইজন

- বাড়ির চাকর
- বাড়ির রাঁধুনী

এবং অন্ত তিনটা

- চাকু
- পিস্তল

- সূক্ষ্ম ও ধারালো লোহার রড

প্রায়োর ডিস্ট্রিবিউশন (Prior Distribution)

Prior বা প্রায়োর হল, অ্যাপ্রিওরিমেশন করার আগে বেসিক কিছু ধারণা নিয়ে কাজ শুরু করতে হয় যেটাকেই প্রায়োর বলে।

এই সমস্যার প্রায়োর হল এটা,

- বাড়ির চাকর এই পরিবারে অনেকদিন ধরে আছে এবং বিশ্বাসভাজন
- আর রাঁধুনীকে বেশিদিন হল নিয়োগ দেয়া হয় নি, এবং তার সম্পর্কে কিছু গুজবও আছে

এই প্রায়োর জ্ঞান দিয়ে আমরা ধারণা করে ফেলতে পারি, রাঁধুনীই আসল খুনি! এবং চাকরের খুনি হওয়ার সম্ভাবনা অনেক কম। ম্যাথেমেটিক্যাল নোটেশনে লিখতে চাইলে এভাবে লিখব,

$$\begin{aligned} P(\text{Culprit} = \text{Servant}) &= 20\% \\ P(\text{Culprit} = \text{Cook}) &= 80\% \end{aligned}$$

এখানে `Culprit` একটা ড্যারিয়েবল যার স্টেট দুইটা হতে পারে, একটি হল `Servant` এবং আরেকটি হল `Cook`। যেহেতু আমরা এখন পর্যন্ত রহস্য সমাধান করতে পারি নি তাই `Culprit` এর আসল স্টেট আমাদের কাছে অজ্ঞান।

প্রব্যাবিলিটির সাম রুল (Sum Rule of Probability)

কিন্তু, $P(\text{Culprit} = \text{Servant}) + P(\text{Culprit} = \text{Cook}) = 100\%$ তারমানে একটা ব্যাপারে আমরা নিশ্চিহ্ন, খুনি `Servant` অথবা `Cook` এদের মধ্যে কেউ একজন। এটা হল **Sum Rule of Probability**।

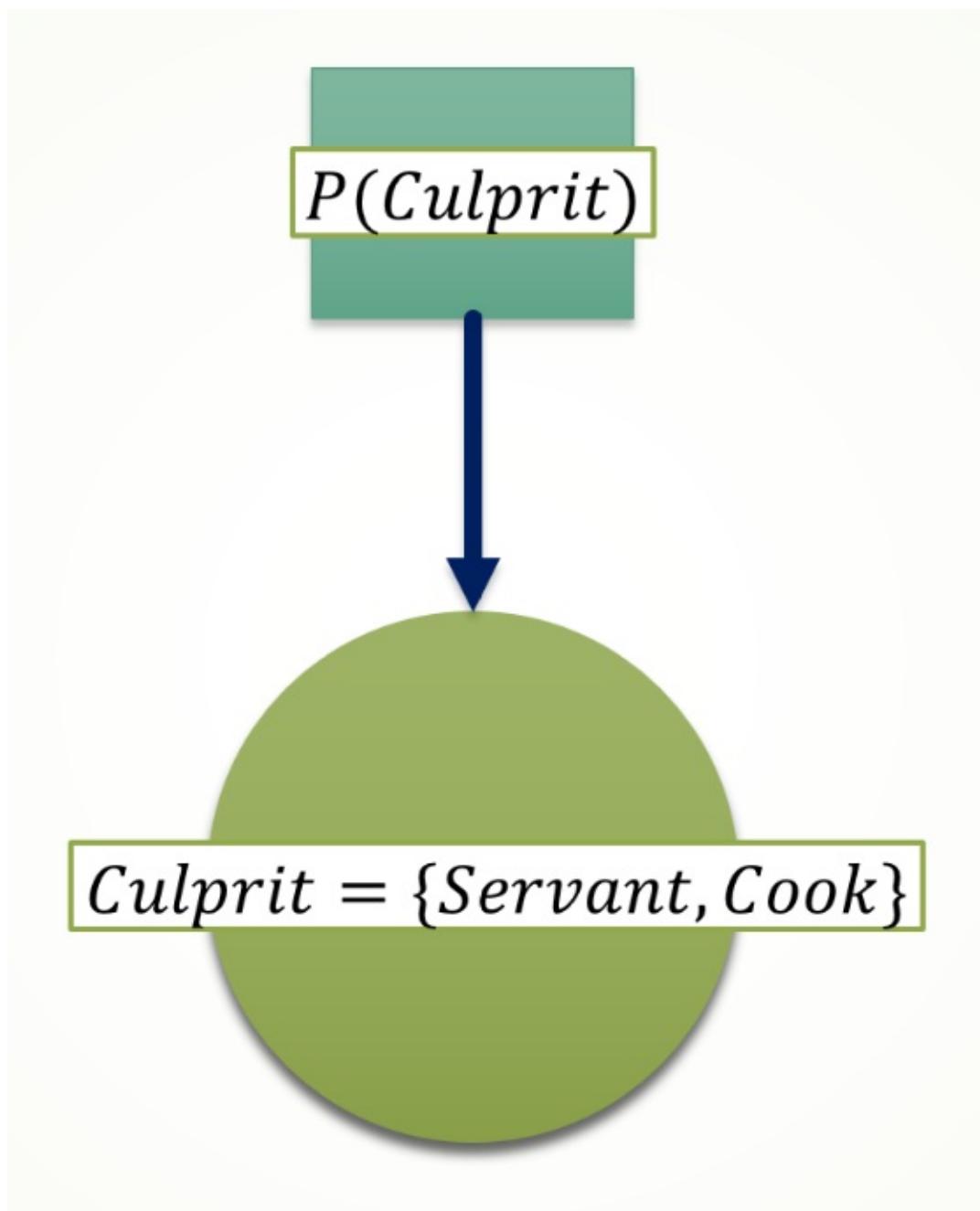
উপরের সমীকরণ দুইটা দিয়ে আসলে প্রব্যাবিলিটি ডিস্ট্রিবিউশন বুঝানো হচ্ছে।

এখানে অনেক প্রশ্ন আসতে পারে, আশি পাসেন্ট কেন? 83% হলে কি সমস্যা? উত্তর হল এটা আমাদের একটা ধারণা, আসল পাসেন্ট কত সেটা আমরা প্রাণ্ট ডেটা থেকে Infer বা প্রেডিষ্ট করব। আমরা ডেটা নিয়ে কাজ শুরুই করিনি।

গ্রাফিকাল নোটেশন প্রব্যাবিলিটির ডিপেন্ডেন্সি ও ইন্ডিপেন্ডেন্সি ভালভাবে দেখাতে পারে। তাই এই নোটেশনগুলোকে গ্রাফিক্যাল রিপ্রেজেন্ট করার চেষ্টা করব। নিচের ছবিটা হল একটা Factor Graph। যেটা সম্পর্কে পরেই বিস্তারিত বলা হবে।

Node: এটা দ্বারা ব্যাখ্যান করা হল `Culprit` এবং যার দুইটা স্টেট হল `Servant`, `Cook`। অথবা, $\text{Culprit} = \{\text{Servant}, \text{Cook}\}$

Square: বর্গ দিয়ে প্রব্যাবিলিটি ডিস্ট্রিবিউশন বুঝায়। অর্থাৎ, $P(\text{Culprit})$ ।



কন্ডিশনাল ডিস্ট্রিবিউশন (Conditional Distribution)

ধরি,

- বাড়ির চাকরের লাইসেন্সড পিস্তল ব্যবহার করার অভিজ্ঞতা আছে। এবং সে পার্সোনাল ড্রয়ারে পিস্তল লকড অবস্থায় রাখে। কিন্তু চাকরের বয়স অনেক বেশি।
- রাঁধুনীর স্বাভাবিকভাবেই চাকুর অ্যান্ডেস আছে।

এখানে কালপ্রিট স্টেট দুইটা, তাই প্রতি স্টেটে আমরা অন্তরের একটা প্রয্যাবিলিটি টেবিল তৈরি করি তাহলে নিচের মত দেখা যাবে।

খুনি 'যদি' রাঁধুনী হয়, তাহলে সে কোন অন্তর ব্যবহার করবে?

Pistol	Knife	Rod
5%	65%	30%

স্বাভাবিকভাবেই রাঁধুনী চাকু দিয়ে খুন করার চেষ্টা করবে যেহেতু তার কাছে সেই অস্ত্রের অ্যাক্সেস পাওয়া সবচেয়ে সহজ। এখানে সব প্রযোবিলিটির মান যোগ করলে 100% হচ্ছে। কারণ, যদি খুনি আসলেই রাঁধুনী হয়ে থাকে, তাহলে ওই তিনধরণের যেকোন একটি দিয়ে সে খুন করেছে। আর যেহেতু ওই তিনটা ছাড়া কোন অপশন নাই তাই তিনটার যেকোন একটা হওয়ার প্রযোবিলিটি শতভাগ।

খুনি 'যদি' চাকর হয়, তাহলে

Pistol	Knife	Rod
80%	10%	10%

এটাও কনভিনিসিং লাগছে, তাই না?

এটা কিন্তু ডেটার ভিত্তিতে আমরা Assume করে নিচ্ছি।

এখন যদি ম্যাথমেটিক্যালি লিখতে যাই,

$$P(\text{Pistol}|\text{Servant}) = 80\%$$

$$P(\text{Knife}|\text{Servant}) = 10\%$$

$$P(\text{Rod}|\text{Servant}) = 10\%$$

সাধারণভাবে আমরা বলি, Probability of Weapon (Example: Knife) given the Culprit (Example: Servant).

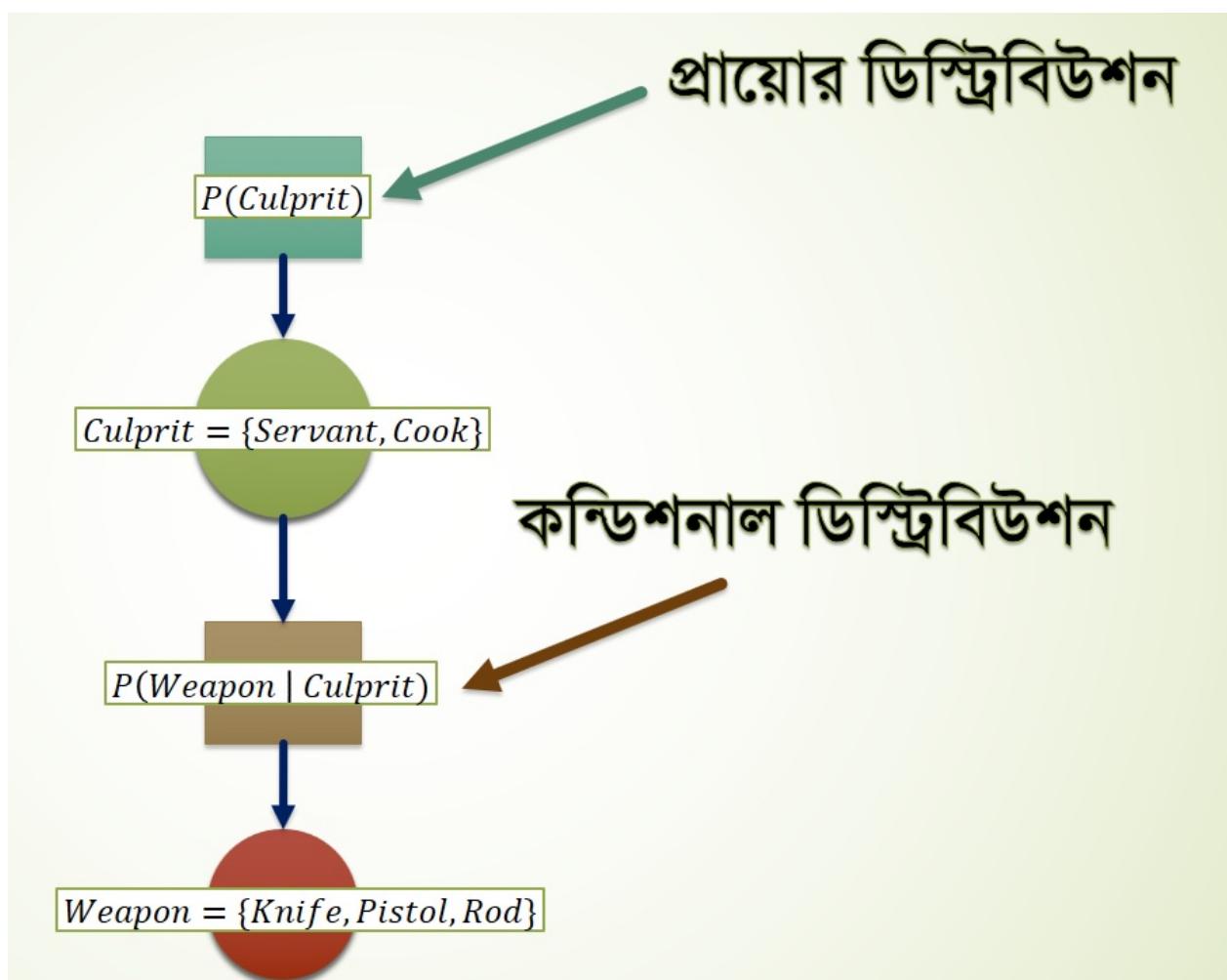
$$P(\text{Weapon}|\text{Culprit})$$

এখানে **Weapon** আরেকটি রংয়ান্ত্র ডায়ারিয়েবল যার স্টেট তিনটা, **Pistol**, **Knife**, **Rod**।

যখনই আমরা কোন একটা ঘটনাকে বাছাই করি, সেটা হয়েছে কিনা, আমরা আসলে তার ডিপেন্ডেন্ট স্টেট গুলোকে জুম করে দেখি। নিচের ছবি দিয়ে পরিষ্কার বোঝা যাবে। [The figure will be added later]

তারমানে **Culprit** রংয়ান্ত্র ডায়ারিয়েবল ইনডিপেন্ডেন্ট, কিন্তু **Weapon** এর স্টেট কোনটা হবে সেটা **Culprit** এর উপর ডিপেন্ডেন্ট।

পরিবর্তিত Factor Graph,



অ্যারো চিহ্ন দিয়ে কোনটা কার উপর নির্ভরশীল বা কন্ডিশনাল ডিপেন্ডেন্ট সেটা বুঝানো হচ্ছে।

জয়েন্ট ডিস্ট্রিবিউশন (Joint Distribution)

জয়েন্ট ডিস্ট্রিবিউশন হল যতগুলো রঞ্জিম ড্যারিয়েবল আছে তাদের জয়েন্টলি যে ডিস্ট্রিবিউশন সেট। যেমন এখানে Culprit আর Weapon এর সম্পর্কিত প্রব্যাবিলিটি ডিস্ট্রিবিউশন ই হবে তাদের জয়েন্ট ডিস্ট্রিবিউশন।

নিচের প্রশ্নের উত্তর আমরা জয়েন্ট ডিস্ট্রিবিউশন দিয়ে দিতে পারব,

| রাঁধুনীর পিস্টল দিয়ে হত্যা করার সম্ভাবনা কত?

এই প্রশ্নের মধ্যে আসলে দুইটা প্রশ্ন আছে। প্রথমটি হল, "খুনির রাঁধুনী হওয়ার সম্ভাবনা কত? (যদি খুনি রাঁধুনী হয়)" দ্বিতীয়টি হল, "খুনির পিস্টল দিয়ে হত্যা করার সম্ভাবনা কত?"।

- প্রথম প্রশ্নের উত্তর $P(\text{Culprit} = \text{Cook}) = 80\%$
- দ্বিতীয় প্রশ্নের উত্তর $P(\text{Weapon} = \text{Pistol} | \text{Culprit} = \text{Cook}) = 5\%$

তাহলে রাঁধুনীর পিস্টল দিয়ে হত্যা করার সম্ভাবনা হচ্ছে,

$$\begin{aligned}
 P(Pistol, Cook) &= P(Pistol \cap Cook) \\
 &= P(Pistol|Cook) \times P(Cook) \\
 &= 5\% \times 80\% \\
 &= 4\%
 \end{aligned}$$

একে Product Rule of Probability বলা হয়ে থাকে। জেনারেল ফর্মে লেখে এভাবে,

$$P(x, y) = P(y|x)P(x)$$

এভাবে আমরা আরও পাঁচটি কম্পিনেশন তৈরি করতে পারি। যেমন, রাঁধুনীর চাকু/রড দিয়ে হত্যা করার সম্ভবনা কত বা চাকরের পিস্টল/চাকু/রড দিয়ে হত্যা করার সম্ভাবনা কত। এই মোট ছয়টি কম্পিনেশনের একটি টেবিল নিচে ক্যালকুলেট করে দেয়া হল।

-	Pistol	Knife	Rod
Cook	4%	52%	24%
Servant	16%	2%	2%

Likelihood এবং Probability এর মধ্যে সূক্ষ্ম তফাত আছে যেটা পরে বলা হবে।

মার্জিনাল ডিস্ট্রিবিউশন (Marginal Distribution)

জয়েন্ট ডিস্ট্রিবিউশন থেকে আমরা সহজেই মার্জিনাল ডিস্ট্রিবিউশন বের করতে পারি। মার্জিনাল ডিস্ট্রিবিউশন মানে হল রংয়ান্ডম ড্যারিয়েবলের কোন একটা স্টেট সিলেক্ট করে, সে স্টেটের যতগুলো প্রব্যাবিলিটির মান জয়েন্ট ডিস্ট্রিবিউশনে আছে তার যোগ করা।

না বুঝা গেলে, আমরা যদি Pistol এর মার্জিনাল ডিস্ট্রিবিউশন বের করতে চাই তাহলে,

$$\begin{aligned}
 P(Pistol) &= P(Pistol|Cook) + P(Pistol|Servant) \\
 &= 4\% + 16\% \\
 &= 20\%
 \end{aligned}$$

Cook এর মার্জিনাল ডিস্ট্রিবিউশন বের করব এভাবে,

$$\begin{aligned}
 P(Cook) &= P(Cook|Pistol) + P(Cook|Knife) + P(Cook|Rod) \\
 &= 4\% + 52\% + 24\% \\
 &= 80\%
 \end{aligned}$$

অর্থাৎ আমরা যেটা Prior ধরেছিলাম সেটাই। বাকি মার্জিনাল ডিস্ট্রিবিউশনও এভাবে বের করতে পারেন।

জেনারেল সূত্র হল,

$$P_X(x) = \sum_y P_{X,Y}(x, y)$$

$P_{X,Y}$ কে আবার কভিশনাল ফর্মে লেখা যায়,

$$P_X(x) = \sum_y P_Y(y) P_{X|Y}(x|y)$$

এটা গেল ডিস্ট্রিট ডেটার ক্ষেত্রে, কটিনিউয়াস ডেটার ক্ষেত্রে সামেশনের জায়গায় ইন্টিগ্রেশন বসবে।

ইন্ডিপেন্ডেন্স

ড্যারিয়েবলের উপর যদি অন্য কোন ড্যারিয়েবল দ্বারা শর্ত আরোপ করলেও সেটার উপরে ওই ড্যারিয়েবলের কোন স্টেট হওয়ার সম্ভাবনা অপরিবর্তিত থাকে তাহলে যে ড্যারিয়েবলের উপর শর্ত আরোপ করা হয়েছিল সেটা ইন্ডিপেন্ডেন্ট বা স্বাধীন ড্যারিয়েবল।

যদি বলা হয়, পর পর দুইটা কয়েন টসে Head উঠার সম্ভাবনা কত? যদি বাইরে বৃষ্টি পড়ে? [ধরি কয়েনটা Fair]

$$\begin{aligned} P(\text{Result} = \text{Head}, \text{Head} | \text{Weather} = \text{Rainy}) &= P(\text{Head} | \text{Rainy}) \times P(\text{Head} | \text{Rainy}) \\ &= P(\text{Head}) \times P(\text{Head}) \\ &= 50\% \times 50\% \\ &= 25\% \end{aligned}$$

বাইরে বৃষ্টি পড়ুক বা না পড়ুক, তাতে টসে হেড উঠবে না টেইল উঠবে তা ডিপেন্ড করে না। তাই আমরা ওটা এড়াতে পারি।

এই বিষয়টা খুব সাধারণ মনে হলেও এটা মারাস্ক শক্তিশালী একটি কনসেপ্ট, পরবর্তী টপিকগুলোতে বারংবার এটা আসবে (Example: Naive Bayes)।

বানুলি ট্রায়াল (Bernoulli Trial)

বানুলি ট্রায়াল হল, কোন একটা ঘটনার ফলাফল যদি শুধু হ্যাঁ/না; 1/0; স্টেট-১/স্টেট-২; সাক্সেস/ফেইলিওর এর মধ্যে সীমাবদ্ধ থাকে তাহলে আমরা তাকে Bernoulli Trial বলব।

উদাহরণ: যেমন কয়েন টস, বৃষ্টি হবে কি না, হ্যাঁ/না ভিত্তিক প্রশ্নের উত্তর ইত্যাদি।

বানুলি ডিস্ট্রিবিউশন (Bernoulli Distribution)

ধরি, `Head` ওঠার সম্ভাবনা যদি 0.5 হয় তাহলে `Tail` উঠার সম্ভাবনা কত? অবশ্যই 0.5। আবার `Head` ওঠার সম্ভাবনা যদি 0.7 হয় তাহলে? `Tail` ওঠার সম্ভাবনা? 0.3! একে ম্যাথেমেটিক্যালি প্রকাশ করা হয় এভাবে,

$$P(\text{Head}|0.5) = 0.5^{\text{Head}} \times (1 - 0.5)^{\text{Tail}}$$

`Head` কে যদি 1 বলি আর `Tail` কে 0, সেক্ষেত্রে `Head` ওঠার সম্ভাবনা,

$$P(1|0.5) = 0.5^1 = 0.5$$

এবং Tail ওঠার সম্ভাবনা,

$$P(0|0.5) = 0.5^0 \times (1 - 0.5)^{1-0} = 0.5$$

গাণিতিকভাবে প্রাক করলে, $x \in \{1, 0\}$ এবং $\theta = [0, 1]$

$$Bern(x|\theta) = \theta^x (1 - \theta)^{1-x}$$

এটাই হল বানুলি ডিস্ট্রিবিউশন! [মনে রাখতে হবে আমি টস করেছি কিন্তু মাত্র একবার!]

বাইনোমিয়াল ডিস্ট্রিবিউশন (Binomial Distribution)

এখন আপনাকে যদি বলা হয়, একটা Fair কয়েন আপনি 10 বার টস করবেন এখন এতে Head 5 বার ওঠার সম্ভাবনা কত? (যদি প্রতি টসের আগের টসের ফলাফল Tail)। 5 টা Head উঠলে বাকি 5 টা অবশ্যই Tail। যেহেতু আমি 10টা টস বিবেচনা করছি তাই আমার Tail ও গণনায় ধরতে হবে।

তাহলে আমি সমস্যা এভাবে মডেল করতে পারি,

$$\begin{aligned} P(\text{Head Count} = 5 | \text{Previous Result} = \text{Tail}) &= P(T) \times P(H|T) \times P(T) \times P(H|T) \times P(T) \times P(H|T) \times P(T) \times P(H|T) \\ &= P(H|T)^5 \times P(T)^5 \\ &= P(H)^5 \times P(T)^5 \\ &= 0.5^5 \times 0.5^5 \times 100 \\ &= 0.09\% \end{aligned}$$

প্রশ্ন আসছে, আমি $P(H|T)$ থেকে কীভাবে $P(H)$ এ গেলাম? আচ্ছা আপনিই চিন্তা করে দেখুন, পরের টসের রেজাল্ট কী হবে সেটা কি আগের টসের রেজাল্টের উপর নির্ভরশীল? অবশ্যই নয়! তাই আমি এই শর্ত সম্পূর্ণ উপেক্ষা করে সাধারণভাবে Head - Tail ওঠার প্রয্যাবিলিটি দিয়ে গুণ দিতে পারি। এটাই ইন্ডিপেন্ডেন্স।

কিন্তু একে ম্যাথেমেটিক্যালি কীভাবে লিখতে পারা যায়? ধরি টস সংখ্যা $N = 10$ এবং রেজাল্ট Head হতে হবে 5 বার তাই $m = 5$ এবং টসে Head হওয়ার সম্ভাবনা $\theta = 0.5$ সুতরাং, 5 বার Head হওয়ার সম্ভাবনা

$$P(m = 5 | N = 10, \theta = 0.5) = 1 \times 0.5^5 \times (1 - 0.5)^{10-5}$$

জেনারেল ফর্মে,

$$Bin(m|N, \theta) = \binom{N}{m} \theta^m (1 - \theta)^{N-m}$$

কিন্তু প্রশ্ন থেকেই যাচ্ছে, $\binom{N}{m}$ আসল কোথা থেকে? এটা দিয়ে বুঝাচ্ছি N টসে m সংখ্যক আউটকাম করতাবে বাছাই করা যায় (Permutation & Combination)। $\binom{10}{5}$ এর মান যেহেতু 1 তাই সেটা দিয়ে গুণ করা হয়েছে আর সবকিছু সিম্পল রাখার জন্য প্রথমে আমি উহু রেখেছি। Maximum Likelihood Estimation এ আমরা এই

কষ্টনেশন নিয়ে কাজ না করলেও সমস্যা নেই। কেন নেই তা পরবর্তী অধ্যায়ের জন্য তোলা থাকল।

বাইনোমিয়াল ডিস্ট্রিবিউশন প্লট

এখন একটু কোড চলে যাওয়া যাক, যদি $N = 10$ এবং $\theta = 0.25$ হয়। তারমানে টসে Head ওঠার সম্ভাবনা 25%, অন্য কথায় কয়েনটা হাইলি বায়াসড। তাহলে তার ডিস্ট্রিবিউশন প্লট কিরকম হবে?

```
import seaborn as sns
import matplotlib.pyplot as plt
import scipy

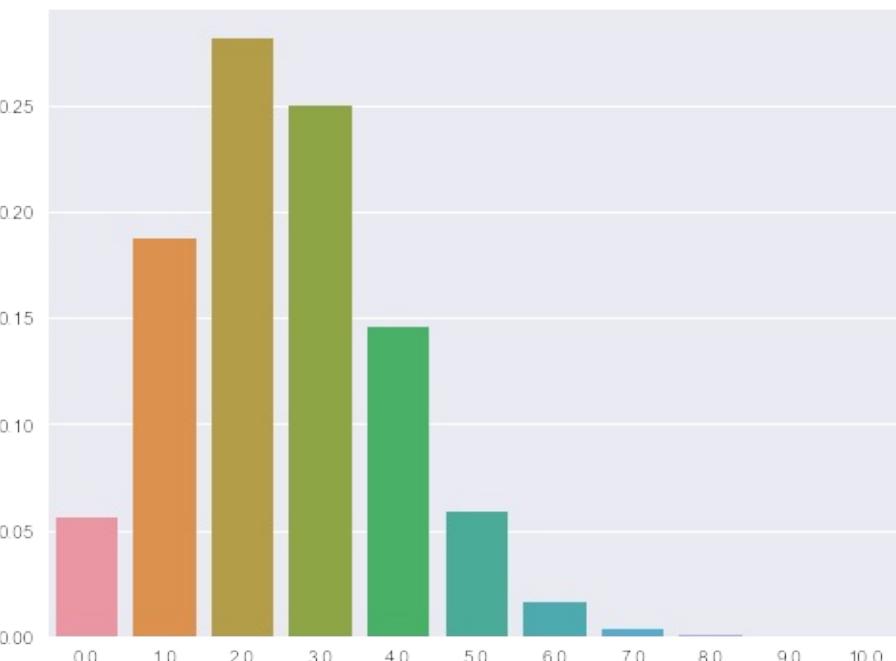
plt.figure(figsize=(8, 6))

theta = 0.25
N = 10

head_counts = scipy.linspace(0, 10, 11) # Number of head counts

pmf = scipy.stats.binom.pmf(x, N, theta)
sns.barplot(head_counts, pmf)
```

আউটপুট



এখানে X এক্সিসে দেখানো হচ্ছে m এর বিভিন্ন মান আৰ Y এক্সিসে দেখানো হচ্ছে সেটা হওয়ার সম্ভাবনা কতটুকু। যেহেতু আমরা এখানে বলেছি, θ এর মান 0.25 তাই $m = 2$ ও এর আশেপাশের বার গুলো আকারে লম্বা, মানে 10 টি টসে $\theta = 0.25$ হলে 2 টি হেড ওঠার সম্ভাবনা বেশি। এভাবে N , m ও θ পরিবর্তন করে আমরা বের করতে পারি কতগুলা টসে কত হেড উঠবে।

পরবর্তী অধ্যায়ে ম্যাঞ্চিমাম লাইকলিহ্ড এস্টিমেশন নিয়ে আলোচনা করা হবে।

Numpy পরিচিতি

দ্রষ্টব্য: এখন থেকে কোডগুলো করা হবে **Python 3** এ, আপনি যদি **Python 2** সেটাপ দিয়ে থাকেন তাহলে একটি ভার্চুয়াল এনভায়রনমেন্ট তৈরি করে **Python 3** সেটাপ দিয়ে দিন, আগের কোডগুলো **Python 3** এ রূপান্তরিত করার প্রক্রিয়া চলছে।

Numpy ইন্সটলেশন

আপনার Numpy না থাকলে কমান্ড উইন্ডো / টার্মিনালে নিচের কোড লিখুন,

```
pip install numpy scipy matplotlib ipython jupyter pandas sympy nose
```

আর আপনি অ্যানাকোড সেটাপ দিয়ে থাকলে আপনার পিসিতে অলবেড়ি Numpy ইন্সটলড আছে। কোন সমস্যা দেখা দিলে [এই ডকুমেন্টেশন দেখুন](#)।

গ্রেডিয়েন্ট ডিসেন্ট আমরা চাইলে একাধিক লুপ অ্যাপ্লাই করে সিঙ্গেল এলিমেন্ট দিয়ে করতে পারি কিন্তু সেটা মোটেও এফিশিয়েন্ট হবে না। মেশিন লার্নিংয়ের জন্য কম্পিউটেশন টাইম কমানোটা খুবই গুরুত্বপূর্ণ। আর সেটা করতে হলে আমাদের অবশ্যই Numpy লাইব্রেরির উপর ভাল দখল থাকতে হবে। ধীরে ধীরে সমস্যা সমাধানের মাধ্যমে Numpy লাইব্রেরির উপরে এমনিতেই দক্ষতা চলে আসবে।

Numpy এ হাতেখড়ি

সায়েন্টিফিক কম্পিউটেশনের জন্য মূলত Numpy ব্যবহার করা হয়। মেশিন লার্নিং সমস্যা গুলোতে হাইডাইমেনশনাল অ্যারে নিয়ে কাজ করতে হয় সেকারণে আমাদের এমন ধরণের টুল দরকার যেটা এই ধরণের হাইডাইমেনশনাল অ্যারে নিয়ে খুবই ফাস্ট কাজ করতে পারে। Numpy হল এমন ধরণের একটি লাইব্রেরি। MATLAB এ আমরা যেভাবে অ্যারে নিয়ে কাজ করে থাকি, Numpy কে আমরা সেক্ষেত্রে Python এর MATLAB ইন্টারফেস বলতে পারি। তবে বেশ কিছু ডিভারিয়েশন আছে।

পুরোপুরি জানার জন্য নামপাইয়ের ডকুমেন্টেশন যথেষ্ট। তবে এখানে আমি গুরুত্বপূর্ণ নিয়ে আলোচনা করব। তাহলে শুরু করা যাক।

অ্যারে (Array)

Numpy Array হল কতগুলো ভ্যালুর গ্রিড। এবং সবগুলো ভ্যালুর টাইপ একই, মানে `float` , `int64` , `int8` ইত্যাদি।

একটা অ্যারের ডাইমেনশন যত তাকে আমরা **Rank** বলে থাকি। যেমন 2 Dimensional Numpy Array কে আমরা বলব `Rank 2 Array`। Numpy এ অ্যারের `Shape` `Integer` এর একটা `Tuple` যেখানে প্রতিটি ডাইমেনশনে কতগুলি এলিমেন্ট আছে সেটা প্রকাশ করে।

নিচের উদাহরণ দেখা যাক,

```
import numpy as np
a = np.array([1, 2, 3])      # Creates a rank 1 array
print (type(a))             # Prints "<class 'numpy.ndarray'>"
print (a.shape)              # Prints "(3,)"
print (a[0], a[1], a[2])    # Prints "1 2 3"

a[0] = 5                    # Change an element of the array
print(a)                     # Prints "[5 2 3]"

b = np.array([[1, 2, 3], [4, 5, 6]]) # Create a rank 2 array
print (b.shape)              # Prints "(2, 3)"
print (b[0, 0], b[0, 1], b[1, 0]) # Prints "1 2 4"
```

Numpy এ কিছু ফাংশনও আছে যেগুলো ব্যবহার করে আমরা নির্দিষ্ট আকারের অ্যারে তৈরি করতে পারি,

```
a = np.zeros((2, 2))      # Create an array of all zeros
print (a)                  # prints "array([[ 0.,  0.],
                           #           [ 0.,  0.]])"
b = np.ones((1, 2))        # Create an array of all ones
print (b)                  # Prints "[[ 1.  1.]]"

c = np.full((2, 2), 7)     # Create a constant array
print (c)                  # Prints "array([[ 7.,  7.],
                           #           [ 7.,  7.]])"

d = np.eye(2)               # Create a 2x2 Identity matrix
print (d)                  # Prints "[[1. 0.
                           #           [0. 1.]]"

e = np.random.random((2, 2)) # Create an array filled with random values
print (e)                  # In my case it printed "array([[ 0.91072458,  0.47086205],
                           #           [ 0.20084157,  0.44929267]])"
```

অ্যারে সম্পর্কে আরও জানতে [এই ডকুমেন্টেশনটি](#) দেখুন।

অ্যারে ইনডেক্সিং (Array Indexing)

বেশ কিছু উপায়ে Numpy অ্যারে ইনডেক্স করা যায়।

স্লাইসিং (Slicing)

পাইথন লিস্ট আমরা যেভাবে স্লাইস করি, সেভাবেই Numpy অ্যারেও স্লাইস করা যায়, Array যেহেতু মাল্টিডাইমেনশনাল হতে পারে সেক্ষেত্রে প্রতিটা ডাইমেনশনের জন্য উল্লেখ করতে হবে কোন ইনডেক্স থেকে কত পর্যন্ত আপনি স্লাইস করতে

```

import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[1 2 3 4]
#  [5 6 7 8]
#  [9 10 11 12]]
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])

# Use slicing to pull out the subarray consisting of the first 2 rows and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]

# A slice of an array is a view into the same data, so modifying it will modify the original array

print(a[0, 1]) # Prints "2"
b[0, 0] = 43 # b[0, 0] is the same piece of data as a[0, 1]
print(a[0, 1]) # Prints "43"

```

চাইলে ইন্টিজার ইন্ডেক্সিং ও স্লাইস ইন্ডেক্সিং মিশিয়েও লেখা যায়। কিন্তু সেটা করলে নতুন অ্যারের Rank ১ করে কমবে, যেমন

```

import numpy as np
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])

# Two way of accessing the data in the middle row of the array
# Mixing integer indexing with slices yields an array of lower rank
# While using only slices yields an array of the same rank as the original array

row_r1 = a[1, :] # Rank 1 view of the second row of a
row_r2 = a[1:2, :] # Rank 2 view of the second row of a

print(row_r1, row_r1.shape) # Prints "[5 6 7 8] (4, )"
print(row_r2, row_r2.shape) # Prints "[[5 6 7 8]] (1, 4)"

# We can make the same distinction when accessing column of an array
col_r1 = a[:, 1]
col_r2 = a[:, 1:2]

print(col_r1, col_r1.shape) # Prints "[2 6 10] (3,)"
print(col_r2, col_r2.shape) # Prints "[[[2
                           #          [6]
                           #          [10]]]] (3, 1)"

```

ইন্টিজার অ্যারে ইন্ডেক্সিং (Integer Array Indexing)

Numpy অ্যারে ইন্টিজার দিয়ে স্লাইসিং করলে নতুন অ্যারেগুলো সবসময়ই আসল অ্যারের সাব অ্যারে হবে। মানে, ইন্টিজার অ্যারে ইন্ডেক্সিং দিয়ে নতুন আরবিটরারি অ্যারে তৈরি করা যায় যে অ্যারের এলিমেন্ট আসবে আসল অ্যারে থেকে।

আমাদের যদি এমন একটা অ্যারে দরকার হয় যার এলিমেন্টগুলো অ্যাসেন্ডিং অর্ডারে থাকবে যেমন `0, 1, 2, 3` তাহলে `np.arange(num)` ফাংশন দিয়ে ওরকম অ্যারে তৈরি করা যায়।

উদাহরণ দেখলে বিষয়টা বুঝা যাবে,

```
import numpy as np
a = np.array([[1, 2], [3, 4], [5, 6]])

# Example of integer array indexing
# The new array will have shape (3, )
print (a[[0, 1, 2], [0, 1, 0]]) # Prints "[1 4 5]"

# Which is equivalent to this one
print (np.array([a[0, 0], a[1, 1], a[2, 0]])) # Prints "[1 4 5]"

# We can also write in this way [Plain old indexing]
print (np.array([a[0][0], a[1][1], a[2][0]]))

# Create sequence of array using `arange` function
sequence = np.arange(4)
print(sequence) # Prints "[0 1 2 3]"
```

আরেকটা উপায়ে ইন্ডেক্সিং করা যায়, যেমন

```

import numpy as np

a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])

print(a)

# Prints
#[[ 1  2  3]
#[ 4  5  6]
#[ 7  8  9]
#[10 11 12]]

# Create an array of indices [we can refer this as selector]
selector = np.array([0, 2, 0, 1])

# Selecting one element from each row using the selector indices
print(a[np.arange(4), selector]) # Prints "[ 1  6  7 11]"

# Mutate one element from each row of 'a' using the indices in b
a[np.arange(4), selector] += 10

print (a)

# Prints
# "[[11  2  3]
#[ 4  5 16]
#[17  8  9]
#[10 21 12]]"

```

বুলিয়ান এক্সপ্রেশন দিয়ে অ্যারে ইন্ডেক্সিং (Boolean Array Indexing)

বুলিয়ান অ্যারে ইন্ডেক্সিং এর মাধ্যমে আমরা বিভিন্ন শর্ত দিয়ে এলিমেন্ট বাছাই করতে পারি। Pandas লাইব্রেরিতেও এই কাজটা করা যায়। উদাহরণের মাধ্যমে দেখা যাক,

```

import numpy as np
a = np.array([[1, 2], [3, 4], [5, 6]])

# Find the elements of 'a' that are bigger than 2
# This returns a numpy array of booleans of the same shape as 'a'
# where each slot of bool_idx tells whether that element of 'a' is > 2
bool_idx = (a > 2)

print(bool_idx)
# Prints
# [[False False]
#  [True True]
#  [True True]]"

print (a[bool_idx]) # Prints "[3 4 5 6]"

# We can reduce all of the statement into a concised single statement
print (a[a > 2]) # Prints "[3 4 5 6]"

```

অনেক সংক্ষেপে এখানে ইন্ডেক্সিং উপস্থাপন করা হয়েছে, আরও ডিটেলস এবং জন্য ডকুমেন্টেশন দেখতে হবে।

ডেটাটাইপ (Datatypes)

অ্যারে তৈরির সময় Numpy অনুমান করার চেষ্টা করে আপনি কোন ডেটাটাইপের অ্যারে তৈরি করছেন। কিন্তু আপনি যদি চান Integer দিয়ে অ্যারে তৈরি করবেন কিন্তু পরে float টাইপের এলিমেন্টও রাখতে হতে পারে তাহলে আপনাকে তার অনুমানকে override করতে হবে, সেজন্য Numpy তে একটা অপশনাল আর্গুমেন্ট আছে। উদাহরণে দেখা যাক,

```

import numpy as np

x = np.array([1, 2])      # Let numpy handle the datatype
print (x.dtype)           # Prints "int32"

x = np.array([1.0, 2.0])    # Again let numpy do it's magic
print (x.dtype)           # Prints "float64"

x = np.array([1, 2], dtype=np.int64)  # Forcing a particular datatype
print(x.dtype)             # Prints "int64"

```

অ্যারে ম্যাথ (Array Math)

বেসিক ম্যাথ

এই টপিকটা খুবই গুরুত্বপূর্ণ। কারণ এটা ব্যবহার করেই আমরা লুপ ব্যবহারের হাত থেকে বাঁচব।

মনে রাখতে হবে, ম্যাথমেটিক্যাল অপারেটর সাধারণত এলিমেন্টওয়াইজ কাজ করে। এবং প্রতিটা অপারেটরের কাজ আবার Numpy এর বিল্টইন ফাংশন করেও করা যায়।

```
import numpy as np

x = np.array([[1, 2], [3, 4]], dtype=np.float64)
y = np.array([[5, 6], [7, 8]], dtype=np.float64)

# Element wise sum; both produce this array
# "[[6.0 8.0]
#  [10.0 12.0]]"
print(x + y)
print(np.add(x, y))

# Element wise subtraction
# "[[-4.0 -4.0]
#  [-4.0 -4.0]]"
print(x - y)
print(np.subtract(x, y))

# Element wise multiplication
# "[[ 5. 12.]
#  [21. 32.]]"
print(x * y)
print(np.multiply(x, y))

# Element wise division
# "[[ 0.2 0.33333333]
#  [ 0.42857143 0.5       ]]"
print(x / y)
print(np.divide(x, y))

# Element wise Square root
# "[[ 1.        1.41421356]
#  [ 1.73205081 2.        ]]"
print(np.sqrt(x))
```

ম্যাট্রিক্স অপারেশন

আমরা আগেই দেখেছিলাম, মেশিন লার্নিং মানেই ম্যাট্রিক্স নিয়ে কাজ কারবার, তাই আমাদের Numpy এর মাধ্যমে Matrix ম্যানিপুলেশন ভালভাবে জানতে হবে। এলিমেন্টওয়াইজ গুণ করে কীভাবে, ম্যাট্রিক্স মাল্টিপ্লিকেশন করে কীভাবে। কোডে হাত দেওয়ার আগে ডট গুণন টা একটু রিভাইজ দেওয়া যাক,

ডেটারের ডট গুণন (Dot Product of vectors)

$\vec{a} = \hat{i} + 2\hat{j} + 3\hat{k}$ এবং $\vec{b} = \hat{i} + 2\hat{j} + 3\hat{k}$ এর ডট প্রদাটি হবে,

$$\vec{a} \cdot \vec{b} = 1 \times 1 + 2 \times 2 + 3 \times 3 = 14$$

ম্যাট্রিক্স আকারে

$$A = [1 \ 2 \ 3]$$

$$B = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$A \cdot B = [1 \ 2 \ 3] \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = 14$$

ম্যাট্রিক্সে ডট গুণন (Dot product of Matrices / Multiplication of Matrices)

কিন্তু যদি ম্যাট্রিক্সের ডট গুণনের কথা চিন্তা করি তাহলে এইরকম হবে,

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$B = [1 \ 2 \ 3]$$

এখন এই দুইটা ম্যাট্রিক্সের ডট গুণন কী হবে? দেখা যাক,

$$A \cdot B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot [1 \ 2 \ 3]$$

$$= [1 \times 1 + 2 \times 2 + 3 \times 3 \quad 4 \times 1 + 5 \times 2 + 6 \times 3 \quad 7 \times 1 + 8 \times 2 + 9 \times 3]$$

$$= [14 \ 32 \ 50]$$

ধরি, C একটি ম্যাট্রিক্স,

$$C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

যদি C কে C এর সাথে ডট গুণন বা ম্যাট্রিক্স মাণিপিলেশন করি তাহলে,

$$C \cdot C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \times 1 + 2 \times 3 & 1 \times 2 + 2 \times 4 \\ 3 \times 1 + 4 \times 3 & 3 \times 2 + 4 \times 4 \end{bmatrix}$$

$$= \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

কিন্তু,

$A \cdot C =$ করা যাবে না,

দুইটা ম্যাট্রিক্সের ডট প্রডাক্টের শর্ত হল, যদি প্রথম ম্যাট্রিক্সের ডাইমেনশন m_1, n_1 হয় এবং দ্বিতীয় ম্যাট্রিক্সের ডাইমেনশন m_2, n_2 হয় তাহলে $n_1 = m_2$ হতে হবে

A এর ডাইমেনশন 3×3 এবং C এর ডাইমেনশন 2×2 তাই এদের মাল্টিপ্লাই করা যাবে না।

এবার কোড দেখা যাক,

```
import numpy as np

x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6], [7, 8]])

v = np.array([9, 10])
w = np.array([11, 12])

# Inner product of vectors
print (v.dot(w))
print np.dot(v, w)

# Matrix/vector product;
print (x.dot(v))
print np.dot(x, v)

# Matrix/ matrix product; both produce the rank 2 array
# [[19 22]
#  [43 50]]
print (x.dot(y))
print (np.dot(x, y))
```

এখন আমাদের যদি সব গুলো এলিমেন্টের যোগফল কিংবা কলামওয়াইজ যোগফল লাগে সেক্ষেত্রে Numpy এর `sum` ফাংশনটি খুব কাজে দেয়।

```
import numpy as np

x = np.array([[1, 2], [3, 4]])

print(np.sum(x)) # Compute sum of all elements; prints "10"

print(np.sum(x, axis=0)) # Compute sum of each column; prints "[4 6]"
print(np.sum(x, axis=1)) # Compute sum of each row; prints "[3 7]"
```

ব্রডকাস্টিং (Broadcasting)

যদি বিভিন্ন শ্রেণীর অ্যারে নিয়ে কাজ করতে হয় সেক্ষেত্রে Numpy এর Broadcasting মেকানিজম খুবই কাজে লাগে। যেমন, আমরা যদি Numpy এর ব্রডকাস্টিং ছাড়া নিচের কাজটা করতে চাই,

```
import numpy as np

x = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
v = np.array([1, 0, 1])

# We will now add the vector 'v' to each row of the matrix 'x'
# Storing the result in the matrix 'y'
y = np.empty_like(x) # Create an empty matrix with the same shape as 'x'

# Add the vector 'v' to each row of the matrix 'x' with an explicit loop
for i in range(4):
    y[i, :] = x[i, :] + v

# Now 'y' is the following
# [[2 2 4]
#  [5 5 7]
#  [8 8 10]
#  [11 11 13]]
print(y)
```

কিন্তু, ম্যাট্রিক্স x যখন অনেক বড় হবে, লুপ দিয়ে এভাবে কম্পিউট করা স্লো হয়ে যাবে। আমরা যদি v এর আরও তিনটা কপি করে রো ওয়াইজ সাজাতে পারি,

$$v = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

তাহলে কিন্তু আমরা সহজেই x এর সাথে যোগ করতে পারব। এই কপি করাটা Numpy এ এভাবে করা যায়,

```

import numpy as np

x = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
v = np.array([1, 0, 1])

# Stacking 4 copies of 'v' on top of each other [4 -> 4 rows, 1 -> 1 rows]
vv = np.tile(v, (4, 1))

print(vv)
# Prints "[[1 0 1]
#           [1 0 1]
#           [1 0 1]
#           [1 0 1]]"

y = x + vv # Adding elementwise

print(y)
# [[2 2 4]
#  [5 5 7]
#  [8 8 10]
#  [11 11 13]]

```

আসলে এত সব কাজ করারও কোন দরকার ছিল না, Numpy এটা নিজেই হ্যান্ডেল করে থাকে, আব এটাই হল Numpy এর Broadcasting

```

import numpy as np

x = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
v = np.array([1, 0, 1])

y = x + v
print (y)
# [[2 2 4]
#  [5 5 7]
#  [8 8 10]
#  [11 11 13]]

```

ব্রডকাস্টিং সম্পর্কে আরও বিস্তারিত জানতে [Numpy User Guide, Release 1.11.0 - Section 3.5.1 \(General Broadcasting Rules\)](#) দেখুন

ব্রডকাস্টিংয়ের অ্যাপ্লিকেশন

```
import numpy as np

## Example 1
# Computing the outer product of vectors
v = np.array([1, 2, 3]) # shape (3, )
w = np.array([4, 5]) # shape (2, )

# To compute an outer product, we first reshape 'v' to be a column vector of shape (3, 1)
# Then we can broadcast it against 'w' to yield an output of shape (3,2)
# Which is the outer product of 'v' and 'w':
# [[ 4  5]
# [ 8 10]
# [12 15]]
print(v.reshape(3, 1) * w)

# Add a vector to each row of a matrix
x = np.array([[1, 2, 3], [4, 5, 6]])
# [[2 4 6]
# [ 5 7 9]]
print(x + v)

## Example 2
# Let's add vector 'w' with 'x' [x.T == x.transpose()]
z = x.T + w

print(z)
# prints
# [[ 5  9]
# [ 6 10]
# [ 7 11]]

# Now we have to transpose it again to revert back to original shape
print(z.T)

# prints
# [[ 5  6  7]
# [ 9 10 11]]
```

Numpy এর বেসিক কিছু অপারেশন দেখানো হল। পরবর্তী পরেই আমরা Numpy লাইব্রেরি ব্যবহার করে ফরমুলা অ্যাপ্লাই করা শুরু করব।

মিনি প্রজেক্ট : হাতে লেখা বাংলা সংখ্যা ডিটেকশন

কাজের ধারা:

কৃত্রিম নিউরনে অনুরণন: পর্ব - ১

ফেলুদা হাতের বইটা সশব্দে বন্ধ করে টক টক দুটো তুড়ি মেরে বিরাট হাই তুলে বলল, 'রোবট'।

আমি জিজ্ঞেস করলাম, 'এতক্ষণ কি তুমি রোবটিক্সের বই পড়ছিলে?'

বইটায় একটা খবরের কাগজের মলাট দেওয়া, তাই নামটা পড়তে পারিনি। এটা জানি যে, ওটা সিধু জ্যাঠার কাছ থেকে ধার করে আনা। সিধু জ্যাঠার খুব বই কেনার বাতিক। সবাইকে ধার দেন না, তবে ফেলুদাকে দেন। ফেলুদাও সিধু জ্যাঠার বই বাড়িতে এনেই আগে সেটায় মলাট দিয়ে নেয়।

একটা চারমিনার ধরিয়ে পর পর দুটো ধোঁয়ার রিং ছেড়ে ফেলুদা বলল, 'রোবটিক্সের বই বলে আলাদা কিছু নেই। সবই মেকানিক্স, কৃত্রিম বুদ্ধিমত্তা আর কট্টোল সিস্টেমের কঞ্চিনেশন। তুই যে ইউটিউবে ভিডিও দেখিস, সেখানে রিকমেন্ডেশন আসে, তোর ইমেইলে আলাদা স্প্যাম ফোন্ডারে মেইল জমা হয়, তুই যে অ্যামাজন থেকে প্রোডাক্ট কিনিস সেখানেও বলে, এটার সাথে ওটা কিনুন কিংবা চিন্তা করে দ্যাখ তোর ফেসবুক অ্যাকাউন্টের People you may know এ বহু পুরাতন বন্ধুর আইডি যাদুমন্ত্রবলে খুঁজে দেয় যার সাথে কিনা গত ৮-৯ বছরে কথাই হয় নি। গুগল সার্চ এত ভাল কাজ করে কীভাবে কখনো ভেবে দেখেছিস? এত যে প্রোগ্রামিং করিস, কখনো এগুলো নিয়ে চিন্তা করেছিস? করিসনি!'

শনিবার সকাল, আমরা দু'জনে আমাদের বাড়ির একতলার বৈঠকখানায় বসে আছি। কাছেই কোথাও পুরাতন বাংলা গানের আওয়াজ ডেসে আসছে। আমার মাথার হাত তিনেক উপর দিয়ে একটা অটোনোমাস কোয়াডকপ্টার উড়ে গেল। আমি ভাবলাম, এটাও তো রোবট হতে পারে!

'কেবল সফটওয়্যার নয়', ফেলুদা বলে চলল, 'এই যে অটোনোমাস কোয়াডকপ্টার উড়ে গেল সেখানেও আছে ইন্টেলিজেন্স।'

ফেলুদার দৌলতে অবশ্য আমার বিভিন্ন রকম ইন্টেলিজেন্স বিশিষ্ট মানুষের সাথে পরিচয় হয়েছে। ফেলুদার ইন্টেলিজেন্স কোন পর্যায়ের সেটাই ভাবছিলাম। ওকে জিজ্ঞেস করাতে বলল, 'নন বেশিক্যাবল হাইপারকনশাস ইন্টেলিজেন্স বলতে পারিস।'

আর আমি তাহলে কী?'

'তোর ইন্টেলিজেন্স লেডেল একটা বিল্ডুর সমান যেটাকে অভিধানে বলে পরিমাণহীন স্থাননির্দেশক চিহ্ন।'

ফেলুদা যতই মজা করুক, ওর সাথে থাকলে আমার ইন্টেলিজেন্স লেডেল বাড়বে বৈ কমবে না। সময়টা এখন ক্লোজড সোর্সের। আর্টিফিশিয়াল ইন্টেলিজেন্স এখন সোনার হরিণ। কোড, রিসার্চ পেপার, ব্লগ পোস্ট স-ব নিষিদ্ধ। ওপেন সোর্স অ্যাস্টিভিটি কোথাও কোথাও হয় বলে জানা গেছে, কিন্তু সেটা খুবই চুপিচুপি। বড় বড় কোম্পানিগুলো একচেটিয়া ব্যবসা শুরু করেছে। কোথাও কোন খিওরি বা কোড লিক হলৈই তাকে গুম করে দেওয়া হচ্ছে। আমি খুবই সাধারণ মানের প্রোগ্রামার, সি কিছুদূর শিখে পাইখন শিখছি মাত্র। পাইখনে গোটাকয়েক সায়েন্টিফিক ক্যালকুলেশন আর স্ট্যান্ডার্ড লাইব্রেরি ছাড়া কিছুই এখন আর পাওয়া যায় না। আগে শুনেছিলাম আর্টিফিশিয়াল ইন্টেলিজেন্স মডেল তৈরি করা ছিল বড়জোর ২ মিনিট এর কাজ। কিন্তু এ নিয়ে আর কোন রিসোর্স বর্তমানে নেই। শুধু সায়েন্স ফিকশনেই সীমাবদ্ধ।'

সবেমাত্র কম্পিউটারের সামনে বসতে যাব এমন সময় দরজার কড়াটা কে যেন সজোড়ে নেড়ে উঠল। দরজা খুলে দেখি ইন্সপেক্টর অমিত বাবু।

অমিত বাবু বললেন, 'তোমার দাদা বাড়িতে আছেন?'

শুনে ফেলুদা সোফা ছেড়ে দরজার দিকে এগিয়ে গেল।

'আরে অমিত বাবু যে, হঠাতে কী মনে করে?'

'আর বলবেন না মি. প্রদোষ মিত্র, সাতসকালেই উক্ত ঘটনার কুলকিনারা না পেলে যা হয় আরকি। আপনাকে তো আমি বিলক্ষণ চিনি, তাই দেবি না করে চলে এলুম।'

ফেলুদা আমার দিকে ফিরে বলল, 'তোপসে, যা তো ভেতরে চায়ের কথা বলে আয়।'

বেয়ারা শৎকরকে চায়ের ফরমায়েশ দিয়ে ফিরতেই শুনতে পেলাম ফেলুদা হাতে একটা কাগজ নিয়ে বলছে,

'হ্ম, বুঝলাম। তাহলে ডাকে যেসব চিঠি এসেছে, সবগুলোতেই এইরকম নম্বর দেয়া? ভদ্রলোক মারা যাওয়ার কত আগে থেকে আসা শুরু করে?'

'হ্যাঁ সবগুলোই আইডেন্টিক্যাল তবে নম্বর আলাদা। তবে একটা ব্যাপার লক্ষ্য করেছি, প্রত্যেকটা চিঠিতে ডিজিট সংখ্যা সমান। চিঠি আসা শুরু করে দিন সাতেক আগে থেকে।', বললেন অমিত বাবু।

'ভদ্রলোকের ব্যাপারে কতটুকু কী জানলেন?'

'বেশি কিছু নয়, ভদ্রলোক নিঃস্তান এবং বিশাল সম্পত্তির মালিক। উনার স্ত্রী গত হয়েছেন বছর সাতেক হল। আশেপাশে কোন আঘাতের নামধার পেলাম না খবর দেওয়ার মত।'

'আর খুনীর ব্যাপারে কিছু জানতে পেরেছেন?'

'নাহ, খুন করার পর ঘটনাস্থল পুরো ভালভাবে ঝিন করা করেছে। যতটুকু বুঝলাম খুনী বেশ প্রফেশনাল।'

'আচ্ছা, চিঠিটা আমার কাছে রাখছি। আপাতত বলতে পারি এটা দামী কাগজে 2B পেনিলে লেখা। আপনি বাকি চিঠি যত দ্রুত সম্ভব পৌছে দেবেন। হ্যাত এটা প্র্যাটিক্যাল জোক ছাড়া কিছুই না। তারপরেও বিষয়টা সিরিয়াসলি নিতে হবে।'

'অনেক ধন্যবাদ মি. মিত্রি। তাহলে আজ আমি উঠি।', বলে চলে গেলেন অমিত বাবু। দরজা বন্ধ করে ফেলুদাকে বললাম,

-'ঘটনা কী ফেলুদা? কাগজে কী লেখা আছে? নতুন কোন রহস্য পেয়ে গেলে নাকি?'

-'সংখ্যাতত্ত্ব!', বলে চুপ মেরে গেল ফেলুদা। বুঝলাম এখন আর ওকে জিজ্ঞেস করে কোন উত্তর পাওয়া যাবে না।

আমি রুমে গিয়ে গুগল সামার অফ কোডের প্রিপারেশন নেয়া শুরু করলাম।

পরে ফেলুদার থেকে জানতে পারলাম, শশাঙ্ক মুখাজী নামের এক ভদ্রলোক তার বাড়িতে রহস্যজনকভাবে খুন হন। ঘর থেকে কিছু চুরি যায় নি। আর তার চেয়েও আশ্চর্যজনক ব্যাপার হল, তার নামে সপ্তাহখানেক আগে থেকে প্রচুর পরিমাণে চিঠি আসা শুরু হয়। সবগুলো চিঠিতেই কতগুলো সংখ্যা লেখা, বাংলায়, প্রায় একই হাতের লেখায়। চিঠির সংখ্যা প্রায় শ'তিনেক হবে। তার তিন কুলে কেউ নেই। প্রতিবেশীদের থেকে জানা গেল, তিনি বেশ নির্বাঙ্কাট একজন মানুষ, তার কোন শক্ত থাকতে পারে এটা কেউ কখনো কল্পনাই করতে পারে নি।

ଫେଲୁଦାକେ ଏହି ବ୍ୟାପାର ନିଯେ ଆର ଘାଁଟାଇ ନି । ଏହି ଘଟନାର ମାସଖାନେକ ପର ଯେଟା ଘଟଳ, ତାର ଜନ୍ୟ ଆମରା କେଟେ ପାରତପକ୍ଷେ ପ୍ରସ୍ତୁତ ଛିଲାମ ନା । ସକାଳ ନଟା ପାଁଚ ବେଜେଛେ କି ବାଜେନି, ଏମନ ସମୟ ଟେଲିଫୋନ ବେଜେ ଉଠିଲ । ଫେଲୁଦା ସୋଫାଯ ବିଷ ପଡ଼ିଛିଲ, ତଡ଼ାକ କରେ ଉଠେ ଟେଲିଫୋନଟା ଧରେ ବଲଲ,

-'ହ୍ୟାଲୋ, ପ୍ରଦୋଷ ମିତ୍ର ବଲଛି ।'

-'ଓ ଅମିତ ବାବୁ, ହ୍ୟାଂ ବଲୁନ ।'

-'କେ କୀ! ଆବାର?'

ତିନବାର ହମ, ପାଁଚବାର ଆଚ୍ଛା, ଦୁଇବାର ଓକେ ବଲେ ଫୋନ ରେଖେ ଦିଲ ।

ଆମାର ଚେହାରା ଦେଖେ ଫେଲୁଦା ବଲଲ, 'ଚ', ଆବାରଓ ଡାକ ରହିଲା!

ବିନ୍ଦୁମାତ୍ର ଦେଇ ନା କରଲେ 'ତୈରି ହେଁ ରଓନା ଦିଲାମ ଫେଲୁଦାର ସାଥେ । ଟ୍ୟାଙ୍କିତେ ଓଠାର ପର ଥେକେ ଫେଲୁଦା ପୁରୋ ଚୁପ । କିଛୁକଣ ପର ପର ଏକଇ କାଯଦାଯ ଆଙ୍ଗୁଳ ମଟକାଛେ । ଏଥନ ଓକେ ଡିସ୍ଟାର୍ କରା ଚଲାବେ ନା । କି ଯେନ ଚିନ୍ତା କରଛେ । ଆମିଓ ଡାବା ଶୁଣ କରଲାମ, ଏଭାବେ ଚିଠି ଦେୟର ମାନେ କୀ? ଖୁନ ତୋ ଅନେକଭାବେଇ କରା ଯାଯା! ନମ୍ବରଗୁଲେ ଦିଯେ ସେ ବୋଯାତେ ଚାହେଟା କୀ? ସେଟା କି କୋନ ସିଙ୍କ୍ରେଟ କୋଡ଼?

ଟ୍ୟାଙ୍କି ଥେକେ ନେମେ ଡାଢ଼ା ଚୁକିଯେ ଫେଲୁଦା ଆର ଆମି ଚୁକଲାମ ଘରେର ଭିତରେ । ଇମ୍ପେଟର ଅମିତ ବାବୁ ଦେଖିତେ ପେଯେ ଏଗିଯେ ଏଲେନ ।

'ଆସୁନ ମି. ମିତିର, ଆପନାର ଜନ୍ୟଇ ଅପେକ୍ଷା କରିଛିଲାମ'

'ଭଦ୍ରଲୋକ କୋଥାଯ ଖୁନ ହନ?'

'ବସବାର ଘରେ ରାତ ଦୁଟୋ ନାଗାଦ, ଫରେନସିକ ଏବ ଲୋକଦେର ଏକଦମ ବିଶ୍ୱାସ କରି ନା । ଆପନାକେ ତାଇ ଚେକ କରାର ଜନ୍ୟ ଡେକେଛି । ଆସୁନ ।'

'ଚଲ ତୋପେ ।', ବଲେ ଫେଲୁଦା ଅମିତବାବୁକେ ଅନୁସରଣ କରତେ ଲାଗଲ ।

ଖୁବହି ସାଧାରଣ ମାନେର ବାଡ଼ି, ଢୋକବାର ପରେଇ ଯେଟା ନଜର କାଡ଼େ ସେଟା ହଲ ବିଶାଳାକାରେର ଝାଡ଼ବାତି । ଦେଖେ କୀ ଯେନ ମନେ ହତେ ଥାକଲ, ବୁଝିତେ ପାରଲାମ ନା ଏମନଟା ଲାଗଛେ କେନ । ଯାଇ ହେବେ ଏକଟା ବାରାଲ୍ଦା ପେରିଯେ ବସବାର ଘରେ ଚୁକଲାମ । ସୋଫାସେଟ, ଟିଭି ଓ ବିହେର ଆଲମାରି ମିଲିଯେ ଖୁବ ସାଧାରଣ ଏକଟି ଘର । ଆଲମାରିତେ ଏକନଜର ଚୋଖ ବୁଲିଯେ ବୁଝଲାମ ବେଶିରଭାଗ ବିଷ ଗଣିତେର ଉପର । ଘରେର ମାଝେ ଏକଟି କାଠେର ଟେବିଲ, ତାର ଉପରେ କିଛୁ କାଗଜ ପେପାରଓୟେଟେର ନିଚେ ଚାପା ଦେଯା ଏବଂ ପାଶେ ଦୁଇଟା ବିଷ । ଦେଖେଇ ଚିନତେ ପାରଲାମ! ଏହି କେଇ ଚିଠି!

'ଭଦ୍ରଲୋକେର ନାମ 'କୈଲାସ ଚୌଧୁରି, ପେଶାଯ ଗଣିତେର ଶିକ୍ଷକ । ଓର କିଛୁ ଛାତ୍ରଦେର ଥେକେ ଜାନତେ ପାରଲାମ ଭଦ୍ରଲୋକ ଡାଲ ପଡ଼ାତେ ପାରେନ ନା । ଏକଟୁ ପାଗଲ ଟାଇପେର । କିନ୍ତୁ ତାଇ ବଲେ ତାଁର ଉପର କାରାଓ କ୍ଷୋଭ ଥାକତେ ପାରେ ସେଟା ଅସ୍ଵାଭାବିକ ।', ବଲଲେନ ଅମିତବାବୁ ।

'ହମ... । ଆଚ୍ଛା ଓନାର କୋନ ଛାତ୍ରେର ସାଥେ କଥା ବଲା ଯାବେ?'

'ଅବଶ୍ୟାଇ, ଆପନି ଚାଇଲେ ଏଥନେଇ ବ୍ୟବଶ୍ଵା କରା ସନ୍ତୋଷ !'

'ଏଥନ ଲାଗବେ ନା, ପରେ ହଲେଓ ଚଲବେ । ଆମାର ପର୍ଯ୍ୟବେକ୍ଷଣ କରା ଶେସ । ଆପଣି ଆପନାର ଫରେନସିକ ଟିମ ଡେକେ କାଜ ଶୁରୁ କରତେ ପାରେନ ।'

'ଦେଖେ କୀ ବୁଝଲେନ ମ୍ର. ମିତିର?'

'ଏଥନି କିଛୁ ବଲତେ ପାରଛି ନା, ତବେ ଚିଠିଗୁଲୋ ପେଲେ ଭାଲ ହତ; ଆଗେର ଚିଠିପତ୍ର ଥେକେ କିଛୁ ଉଦ୍ଧାର କରତେ ପାରିନି ତବେ ମନେ ହ୍ୟ ଏବ୍ରଟା ପେଲେ କିଛୁ ଆଲୋକପାତ କରା ଯେତ ।'

'ଚିଠି ଆମି ଆପନାକେ କାଲକେର ମଧ୍ୟେ ପାଠିଯେ ଦିଚ୍ଛି ।'

'ଧନ୍ୟବାଦ ଅମିତବାବୁ', ବଲେ ଫେଲୁଦା ଆରାଓ ଏକବାର ପ୍ରତିଟା ଘର ଭାଲଭାବେ ପରୀକ୍ଷା କରେ ବଲଲ 'ଚଲ ତୋପସେ, ଏଥାନେର କାଜ ଶେସ । ବାକିଟା ଠାଣା ମାଥାଯ ଚିତ୍ରା କରେ କରତେ ହବେ ।'

ବାଡ଼ି ଫିରେଇ ଫେଲୁଦା ତାର ବିଖ୍ୟାତ ସବୁଜ ଖାତାଯ ହିବିଜିବି ଲେଖା ଶୁରୁ କରଲ । ଆମି ମନେର ଅଜାନ୍ତେଇ ଉତ୍ସଖୁଶ କରଛିଲାମ, ଫେଲୁଦା ସେଟା ଲକ୍ଷ୍ୟ କରେ ବଲଲ, 'ତିନଟା ପ୍ରମ୍ବ ।'

'ମାନେ?'

'ଏକଟା ଶେସ!'

ମନେ ମନେ ବଲଲାମ,'ଏଇ ବେ, ଟ୍ରିକଟା ବୋଝା ଉଚିତ ଛିଲ!', ଏକରାଶ ବିରକ୍ତି ବିନ୍ଦୁମାତ୍ର ନା ପ୍ରକାଶ କରେ ବଲଲାମ 'ଚିଠିର ନମ୍ବରଗୁଲୋର ମାନେ କୀ?'

'ଏଥନୋ ଜାନିନା'

'ତୁମି ଦ୍ୱିତୀୟବାର ଘର ପରୀକ୍ଷା କରତେ ଗେଲେ କେନ?'

'ପ୍ରଥମବାର ଆରେକଟୁ ହଲେଇ ମିସ କରେ ଗେସ୍‌ଲାମ । ଟେବିଲେର ଉପରେ ଦ୍ୱିତୀୟ ବିଷିତେ ଏକଟା ବୁକମାର୍କ ଛିଲ, ଚେକ କରତେଇ ଏକଟା ଫୋନ ନାମ୍ବର ପେଯେ ଗେଲାମ ।'

'ଆମିତବାବୁକେ ଜାନାଓ ନି କେନ ତାହଲେ?', ଅବାକ ହ୍ୟ ଜିଜ୍ଞେସ କରଲାମ ।

'ଉହଁ ଆର ନଯ, ପରେ ଜାନତେ ପାରବି!', ବଲେ ଟକାସ କରେ ମାଥାଯ ଏକଟା ଛୋଟ ଗାଁଟା ଦିଯେ ତାରପର ଚାରମିନାର ଧରିଯେ ଫେଲୁଦା ନିଜେର ଘରେ ଗିଯେ ଦରଜା ଆଟକିଯେ ଦିଲ । ଏଥନ ଓର ଚିତ୍ରା କରାର ପାଲା, ଆଶା କରଛି ଜଳଦି କିଛୁ ଜାନତେ ପାରବ ଏଇ ଜଟିଲ ରହସ୍ୟ ସମ୍ପର୍କେ ।

ପରଦିନ ସକାଳେ ଘୁମ ଥେକେ ଉଠେ ଦେଖିଲାମ ଫେଲୁଦା ବାଡ଼ି ନେଇ । କୋଥାଓ ବେରିଯେଛେ । ଆମି ବେର ହବ କିନା ଭାବଛି ଏମନ ସମୟ କଲିଂ ବେଲ ଏର ଆଓଯାଜ ଶୁନତେ ପେଲାମ । ଦରଜା ଖୁଲେ ଦେଖି ହତେ ଦୁଇଟା ପ୍ଲାକେଟ ନିଯେ ଲାଲମୋହନବାବୁ ହାସିମୁଖେ ଦାଁଡିଯେ ଆଛେନ ।

'ତପେଶବାବୁ, ଫେଲୁବାବୁ ଆଛେନ ନାକି?'

'ନା ଦାଦା ନେଇ, ଆପଣି ଭିତରେ ଆସୁନ । କୋନ ଭାଲ ଖବର ନିଯେ ଏଲେନ ନାକି?'

'হ্যাঁ, তোমার দাদার গুণে আমার বইয়ের ভুলটুল তো অনেকটাই কমে এল তাই ভাবলাম ইয়ে মানে ওনাকে একটা ট্রিভিউট দি। তা কোন নতুন রহস্য টহস্য পেয়েছেন নাকি?'

'লালমোহনবাবু, ভাল কোন সংবাদ নিয়ে এসেছেন মনে হচ্ছে?', কোথেকে ফেলুদা হাজির হয়ে বলল।

'আরে বলবেন না মশাই, আমার বই নিয়ে তো ফিল্ম বানানোর প্রস্তাব এসেছে। ফেলু মিতিৰেৱে পৰিচৰ্যায় প্ৰথৰ রুদ্র দিনে দিনে আৱও প্ৰথৰ হয়ে উঠছে। কিন্তু আপনি ঘৰে ঢোকাৰ আগেই বুঝলেন কীভাবে?'

'সহজ, আপনি কোন ভাল সংবাদ নিয়ে আসলে এই ব্র্যান্ডের বড়ি স্প্ৰে মেখে এবং সুস্থাদু খাবাৰ সমেত আসেন, দুইটাৱই গুৰু এখনও বিদ্যমান। আৱ তাছাড়া নতুন পাঞ্জাবি পৰেছেন; গলাৰ কাছটাৰ বোতাম লাগাতে বেশ কসৱৎ কৰতে হয়েছে নিশ্চয়ই, তাই অনুমান কৰতে একটু কষ্ট হয় নি।'

'বাহ বাহ, আপনাৰ জবাৰ নেই। যেটা বলছিলাম, আপনি তো এইসময়ে সাধাৱণত বেৱ হন না; কোন নতুন কেস হাতে নিয়েছেন নাকি?', বললেন লালমোহনবাবু।

'আপনাকে সব বলব তবে তাৰ আগে বলুন আপনাৰ গাড়িটা আজকেৰ দিনেৰ জন্য পাওয়া যাবে কিনা?'

'বিলক্ষণ, ইয়ে মানে, আমি তো অলয়েজ অ্যাট ইওৰ সার্ভিস! শুধু বলে ফেলুন কোথায় যেতে হবে।'

'আপনি নিচে গিয়ে গাড়ি বেড়ি কৰুন আমি আসছি', বলে আমাৰ দিকে ফিরে বলল, 'তুই তৈৰি হয়ে লালমোহনবাবুৰ সাথে গাড়িতে উঠে বস।', বলেই ফেলুদা দ্রুত নিজেৰ ঘৰে চলে গেল। আমি অগত্যা ফেলুদাৰ কথামত নিজেৰ ঘৰে গিয়ে তৈৰি হয়ে লালমোহনবাবুৰ গাড়িতে উঠলাম। মিনিট পাঁচেক পৰ দেখি হঠাৎ অচেনা এক মধ্যবয়স্ক লোক গাড়িতে উঠেই নাঁকিসুৰে বলল, 'চলুন লালমোহনবাবু'। আমি চমকে উঠাৰ পৰমুহূৰ্তেই বুঝলাম, বড় জুলপিওয়ালা, কাঁচাপাকাচুল বিশিষ্ট নাঁকিসুৱেৰ কঠুন্ধাৰী লোকটি আৱ কেউ নন, ফেলুদা।

'তুই হলি আমাৰ দুঃসম্পর্কেৰ আঘৰীয় এবং শান্তিশিষ্ট লেজবিশিষ্ট একটা ছেলে। বেশি কথা বলেছ কী গাঁটা খেয়েছ। প্ৰশ্ন কৰবি না একদম, শুধু আমাৰ কথাৰ সাথে তাল মেলাবি। বুঝেছিস?', বলে মাথায় একটা টোকা দিয়ে বলল, 'এটা হল অৰ্জিনাল গাঁটাৰ ডেমো ভাৰ্সন।'

'বুঝেছি', বলে বাগে গজগজ কৰতে থাকলাম। আমি যেই তিমিৰে ছিলাম সেই তিমিৰেই আছি, ফেলুদা আগে আমাকে ওৱ কেসেৰ ব্যাপাৰে সবসময় আপডেটেড রাখত। হঠাৎ কী হল বুঝতে পাৱলাম না।

যাওয়াৰ পথে আমাৰ জানা অংশটুকুই ফেলুদা লালমোহনবাবুকে জানিয়ে রাখল। লালমোহনবাবু বাব কতক 'হাইলি সাসপিশাস' বলে চুপ মেৰে গেলেন।

গাড়ি থামল একটা ছোটখাট কম্পিউটাৱেৰ দোকানেৰ সামনে, ধুলোমাথা সাইনবোৰ্ডে লেখা ইলেক্ট্ৰোস্টলার্স বিসাৰ্চ অ্যান্ড ডেডেলপমেন্ট সোসাইটি। মানে বুঝলাম না, এৱা কি বলতে চায়? একই সাথে দোকান এবং R&D ?

দোকানে চুক্তেই স্থূলকায় একজন লোক এগিয়ে এসে বললেন,

'কী চাই?'

ফেলুদা বলল, 'আঁমি জয়নারায়ণ বাগচি বলচি। আমিই আপনাকে ফোন কৰেছিলাম আমাৰ ভাইপোৰ ব্যাপাৰে।'

'আচ্ছা, আপনাৰ ভাইপো ডিজিটাল ইমেজ প্ৰসেসিং শিখতে চায়, তাইতো?'

'ହଁ ହଁ, ଆଁପନାର ମେମରି ଦେକଟି ଖୁବ ଇଯେ କି ବଲେ ଯେନ...'

'ଶାର୍ପ?'

'ଶାର୍ପ; ଶାର୍ପ ବାଇଟ', ବାଚା ଛେଲେର ମତ ହାତତାଳି ଦିଯେ ଉଠିଲ ଫେଲୁଦା । ବଲଲ, 'ଇଯେ ମାନେ, ଆଁପନାର ନାମଟା ଜାନା ହ୍ୟ ନି । ଆବ ଫି ଏବ ବ୍ୟାପାରେଓ କିଛୁ ବଁଲେନନି ।'

'ଆସଲେ ଏଟା ନିର୍ଭର କରେ ସେ ଆଗେ ଥେକେ କଟ୍ଟକୁ ଜାନେ ତାର ଉପରେ । ତାର ଯଦି ବେସିକ ପ୍ରୋଗ୍ରାମିଂ ସ୍କିଲ ଥାକେ ତାହଲେ ଆମରା ସେଟା ନତୁନ କରେ ଶେଖାଇ ନା, ସେଜନ୍ୟ ଫି କମ ଦିଲେଓ ଚଲେ । ଆମି ଆଗେ ଏକଟୁ ଚେକ କରେ ଦେଖବ ଆପନାର ଡାଇପୋର ଜ୍ଞାନ କଟ୍ଟକୁ । ଆବ ଆମାର ନାମ ମାନସ, ଆମି ଏହି ଦୋକାନେର ଏକଜନ କର୍ମଚାରୀ ।'

ତାରପର ଉନି ଆମାର ଦିକେ ଫିରେ ବଲଲେନ,

'ଏହି କମ୍ପ୍ଯୁଟଟାରେ ସାମନେ ବସୋ । ତୋମାର ନାମ କୀ?', ଉତ୍ତର ଦେଯାର ପର ବଲଲେନ, 'ଆଜ୍ଞା ତପେଶରଙ୍ଗନ, ତୁମି କି କି ପ୍ରୋଗ୍ରାମିଂ ଲ୍ୟାଙ୍ଗ୍ୟେଜ ପାରୋ?'

ବଲଲାମ, 'C++ ଏବଂ Python କିଛୁଟା ପାରି'

'ଗୁଡ ଗୁଡ, ପାଇଥନେର ବେସିକ ଜାନଲେଇ ଚଲବେ । ସବକିଛୁ ସିମ୍ପଲ ରାଖାର ଜନ୍ୟ ଆମି ତୋମାକେ ପାଇଥନେଇ ଇମେଜ ପ୍ରୋସିଂ ଶେଖବ । କେମନ?'

ତିନଘଟା କ୍ଲାଶ କରାର ପର ମାଥାଟା ଡୋଁ ଡୋଁ କରଛିଲ । ମାଥାମୁଣ୍ଡ କିଛୁଟି ବୁଝଛି ନା ଯେ, ଫେଲୁଦା କେନ ଆମାକେ ଏହି ଧରଣେର କୋର୍ସ କରିଯେ ନିଚ୍ଛେ । ଅବଶ୍ୟ ଜିନିସଟା ବେଶ ଇନ୍ଟାରେଷ୍ଟିଂ । ତାବଲାମ ବାଡ଼ି ଫିରେଇ ଜିନିସଗୁଲୋ ପ୍ରୟୋଗ କରେ ଦେଖବ ।

ଦୋକାନ ଥେକେ ବେର ହ୍ୟେ ଦେଖିଲାମ, ଫେଲୁଦା ହାସଛେ । ବଲଲ, 'ତୋ ଉପର ଅନେକ ପ୍ରେଶାର ଗେଲ । ଆବ ନଯ, ଯେତେ ଯେତେ ସବ ବଲାଛି । ଗାଡ଼ିତେ ଓଠ ।'

ଫେଲୁଦାର ଆବ ଆମାର କଥୋପକଥନେର ସାରମର୍ମ ମୋଟାମୁଟି ଏହି,

ବହିଯେର ବୁକମାର୍କେର ଭେତରେ ଏହି ଦୋକାନେର ଫୋନ ନାସାର ଫେଲୁଦା ପାଯ । ଖୋଜ ଖବର ନିତେଇ ଜାନା ଗେଲ ଏହି ଭଦ୍ରଲୋକ କୈଲାସ ଚୌଧୁରିକେ ଚିନିତ ଏବଂ ତାର ବାଡ଼ିତେ ଯାତାଯାତ ଛିଲ । ଭଦ୍ରଲୋକେର ସମ୍ପର୍କେ ଅନଲାଇନେ ଭାଲମତ ଖୋଜ ନିଯେ ତେମନ କିଛୁ ଜାନା ଯାଯ ନି । ଏତେଇ ଫେଲୁଦାର ସନ୍ଦେହ ହ୍ୟ, ସେ ଏକଜନ କୋର୍ସ ପରିଚାଳକ କିନ୍ତୁ ଦୋକାନ ଅଧିକାଂଶ ସମୟ ଫାଁକା ଥାକେ ଏବଂ ଏହି ଦୋକାନେର ନାମ ଖୁବ କମ ଲୋକେଇ ଜାନେ । ଯେହେତୁ ପ୍ରମାଣ ନେଇ ତାଇ ସରାସରି ଜିଜାସାବାଦ ସନ୍ଦେହେର ଉତ୍ୱର୍କ କରବେ ଏବଂ ଉନି ପାଲିଯେଓ ଯେତେ ପାରେନ । ସେଇ କାରଣେଇ ଫେଲୁଦା ଛନ୍ଦବେଶ ନିଯେ ତାଁ ସାଥେ ଦେଖା କରେ । ଆବ ମି. ମାନସେର କାଛେ ଗୁରୁତ୍ୱପୂର୍ଣ୍ଣ ତଥ୍ ଥାକା ଅସ୍ଵାଭାବିକ କିଛୁ ନା । ସେଟା ବେର କରାର ପାଶାପାଶି ସେଇ ଚିଠିପତ୍ରେର ସଂଖ୍ୟା ବିଶ୍ଵେଷଣ କରାର ଜନ୍ୟରେ ଡିଜିଟାଲ ଇମେଜ ପ୍ରୋସିଂ୍ୟେର ଜ୍ଞାନ ଥାକତେ ହବେ । ତାଇ ଫେଲୁଦା ଆମାକେ ଦିଯେ କୋର୍ସ କରିଯେ ନିଚ୍ଛେ, ଯାତେ ଏକ ଚିଲେ ଦୁଇ ପାଥି ମରେ ।

ବାଡ଼ି ଫେରାର ପର ଫେଲୁଦା ଏକଟା [SD Card Adapter](#) ଧରିଯେ ଦିଯେ ବଲଲ, 'ନେ ଏଇଥାନେ ଚିଠିପତ୍ରେର ସକଳ ନସ୍ବରେ ଛବି ତୋଳା ଆଛେ । ତୁଇ ଯା ଶିଖେଛିସ ତା ଅୟାଖାଇ କରେ ଡିଜିଟାଲ ଏଡିଟ୍ୟାବଲ ଡିଜିଟ ବାନାତେ ପାରିସ କିନା ଦ୍ୟାଖ । ଆମାର ଯଦୁର ମନେ ହ୍ୟ, ଏହି ନସ୍ବରଗୁଲୋତେଇ ରହସ୍ୟେର ଚାବି ଲୁକିଯେ ଆଛେ । କିନ୍ତୁ ଏତ ଏତ ନସ୍ବର ଖାତା-କଲମେ ବିଶ୍ଵେଷଣ କରତେ କରତେ ଆରେକଟା ଖୁନ ହ୍ୟେ ଯେତେ ପାରେ । ତାଇ ଆମାଦେର ଆପାତତ ଏମନ ଏକଟି ସିସ୍ଟେମ ବାନାତେ ହବେ ଯେଟା ଇମେଜ ଥେକେ ଏଡିଟ୍ୟାବଲ ଡିଜିଟେ ରୂପାନ୍ତର କରତେ ପାରେ । ଏବଂ ଏହି କାଜଟା ଆମି ତୋକେ ଦିଲାମ ।'

'ଆଜ୍ଞା ଫେଲୁଦା, ଆମି ଚେଷ୍ଟା କରେ ଦେଖିଛି !', ବଲେ ନିଜେର ସବେ ଆସଲାମ । ବିଛାନାଯ ଶରୀର ଏଲିଯେ ଦିଯେ ଚିତ୍ରା କରତେ ଥାକଲାମ, ଏଠା କୀଭାବେ ସନ୍ତୋଷ ? ଆମାର ନିଜେରିଟି ବାଜେ ହତେର ଲେଖା ଦେଖିଲେ ଚିନତେ ଅସୁବିଧା ହ୍ୟ, ସେଥାନେ ଏମନ ସିସ୍ଟେମ କୀଭାବେ ବାନାବ ? ଆଜ୍ଞା ବରଂ ଏକ କାଜ କରି । ଛବିଗୁଲୋର କୀ ଅବଶ୍ୟା ସେଟା ଏକଟୁ ଦେଖେ ନେଇ ।

କମ୍ପ୍ୟୁଟାର ଚାଲୁ କରେ SD Card Adapter Reader ଦିଯେ କାର୍ଡଟା ରିଡ କରା ଶୁଳ୍କ କରି । ଫୋଲ୍ଡାରେ ଢୁକେ କଯେଟା ଇମେଜ ଓପେନ କରି ।

୦
୧
୨
୩
୪
୫
୬
୭
୮
୯

ଚିତ୍ରା ବିଷୟ !

ଓପେନ କରା ଫୋଲ୍ଡାରେ ଏକଟା ଫିଲିପଟ ।

	bn00000.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00001.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00002.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00003.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00004.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00005.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00006.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00007.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00008.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00009.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00010.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00011.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00012.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB
	bn00013.bmp	Type: BMP File Dimensions: 32 x 32	Size: 3.05 KB

তবে একদিক থেকে মুক্তি পেলাম। প্রতিটা ইমেজ একই আকারের অর্থাৎ 32×32 । তাই এদের প্রসেস করতে সুবিধা হবে। নইলে আমার নিজের এই ছবিগুলোকে বিসাইজ করে নিতে হত।

প্রতিটা ফাইলের শেষের ডিজিট বলে দিচ্ছে ইমেজটা কোন ডিজিটের। ফেলুদার আসলেই জবাব নেই।

আচ্ছা একটা কাজ করা যায় না? প্রতিটা ১ এবং ০ তে গড়ে কতগুলো সাদা এবং কালো পিঙ্কেল আছে, সেগুলো গণনার ভিত্তিতে যদি এদের ভাগ করতে পারি তাহলে তো দুইটা সংখ্যার মধ্যে পার্থক্য ধরতে পারব!

আগেই একগাদা চিন্তা করার চেয়ে কাজে হাত দিয়ে তারপর চিন্তা করা আমার পক্ষে যুক্তিমুক্ত মনে হল, তাই Jupyter Notebook ওপেন করে কোড লেখা শুরু করলাম।

ইমেজ লোড করার কোড লেখা শুরু করি

যেহেতু ইমেজ ব্যবহার লোড করে পরীক্ষা করতে হবে, তাই ভাবলাম এটা করার জন্য একটা ফাংশন লিখি, যেটা কল করলে অটোমেটিক প্রয়োজনমাফিক ইমেজ লোড হবে। এই অংশটুকু আমার পক্ষে করা অত্যন্ত সহজ।

```
from scipy.ndimage import imread
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
import numpy as np

image_path = "./BengaliBMP/"
image_count = 6000
%matplotlib inline
images = [image_path + 'bn' + str(num).zfill(5) + '.bmp' for num in range(image_count)]
]
```

ফেলুদার দেওয়া ইমেজগুলো **RGB** ছিল অর্থাৎ **Red**, **Green**, **Blue** এই তিন চ্যানেলের সমষ্টি। এই রকম আরও অনেক চ্যানেল আছে, যেমন **CMYK**, **Grayscale**, **Binary** ইত্যাদি। আমরা যেভাবে ছবি দেখি কম্পিউটার সেভাবে দেখে না। ও শুধু চেনে নষ্ট। **RGB** হল মৌলিক তিনটা রং, যা দিয়ে বাকি সব রং তৈরি করা যায়। তাই **RGB** চ্যানেলের ইমেজ সাধারণত রঙিন হয়।

54	58	255	8	0
45	0	78	51	100
85	47	34	185	207
22	20	148	52	24
52	36	250	74	214
	158	0	78	51
		72	74	136
			247	255
				74

এইভাবে কম্পিউটারে ম্যাট্রিক্স আকারে সাজানো থাকে **RGB** চ্যানেলের ইমেজ। আমরা একে বলতে পারি ৩ লেয়ারের **2D Matrix Array**। **Grayscale** এবং **Binary** ইমেজের ক্ষেত্রে এই চ্যানেলের সংখ্যা হ্য মাত্র একটা।

ফেলুদার দেয়া ছবিগুলোর সাইজ ছিল 32×32 এবং **RGB**। এর মানে হচ্ছে একেকটা 2D Matrix অ্যারের Row এবং Column 32 টা করে। আমরা যদি এর শেপ লিখতে চাই তাহলে লিখব, $32 \times 32 \times 3$ আকারে।

আমি এখন প্রোগ্রাম করে এই তিনটা চ্যানেল থেকে কমিয়ে একটা চ্যানেলে আনব বা গ্রেস্কেলে আনব। আমি যদি এই কনভার্সন না করি তাহলে আমার $32 \times 32 \times 3 = 3072$ সংখ্যক পিঙ্কেল নিয়ে কাজ করতে হত, গ্রেস্কেলে নিয়ে আসলে আমার $32 \times 32 \times 1 = 1024$ টা পিঙ্কেল নিয়ে কাজ করতে হবে যা তুলনামূলক সহজ কাজ।

এমন একটা ফাংশন এখন লিখতে হবে যেখানে আমি ডিজিট ইনপুট দিলে শুধু ওঁ ডিজিটের ইমেজ প্যাথ দেয়। কিছুক্ষণ চিন্তা করে ঝটপট লিখে ফেললাম,

```
def load_digit(digit, ip=image_path, ic=image_count):
    return [ip + 'bn' + str(num).zfill(5) + '.bmp' for num in range(ic) if str(num)[-1] == str(digit)]
```

একটু টেস্ট করা যাক ঠিকঠাক কাজ করছে কিনা,

```
ones = load_digit(1)
for one in ones:
    print(one)
```

আউটপুট আসল,

```
./BengaliBMP/bn00001.bmp
./BengaliBMP/bn00011.bmp
./BengaliBMP/bn00021.bmp
./BengaliBMP/bn00031.bmp
./BengaliBMP/bn00041.bmp
./BengaliBMP/bn00051.bmp
./BengaliBMP/bn00061.bmp
./BengaliBMP/bn00071.bmp
.....
```

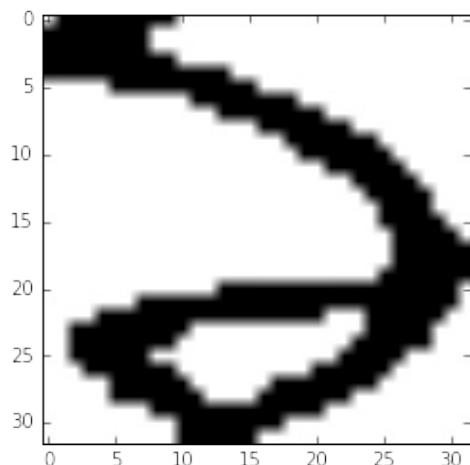
আমি ইমেজ লোড করার সাধারণ কোডের সাথে ছোট একটি শর্ত জুড়ে দিলাম, যেটা কিনা চেক করে নম্বরের শেষের ডিজিট আগুমেন্টে পাস করা ডিজিটের সাথে মেলে কিনা, যদি মিলে যায় তাহলে আমি সেই পথটা রেখে দেব।

ইমেজ শো করি

ইমেজ ঠিকঠাক লোড হচ্ছে কিনা সেটা চেক করতে হবে,

```
one_img = imread(ones[0])
plt.imshow(one_img)
```

ঠিকঠাক লোড হল,



ইমেজ চ্যানেল চেক করি

এটা আসলেই RGB ইমেজ কিনা সেটা চেক করা দরকার,

```
print(one_img.shape)
```

আউটপুট

(32, 32, 3)

হাঁ যেটা বলছিলাম, সেটাই।

ইমেজের চ্যানেল কনভার্ট করি ও চ্যানেল চেক করি

```
one_img_gray = rgb2gray(one_img)
print(one_img_gray.shape)
```

আউটপুট

(32, 32)

এর মানে হল এর চ্যানেল একটাই।

ইমেজের পিক্সেল কাউন্ট করতে যাব এমন সময় ফেলুদা ডাক দিল, 'তোপসে খেতে চলে আয়'।

কখন যে কিদে পেয়ে গিয়েছিল টেরও পাইনি। ফেলুদার ডাকে খেয়াল হল। সব ফাইলগুলো সেভ করে ডাইনিং রুমে
গিয়ে খেতে বসার সময় ফেলুদা বলল,

'সিধু জ্যাঠার কাছে একবার যেতে হবে। ওপেনসোর্স অ্যাষ্টিভিটি নিয়ে কোন সংবাদ আছে কিনা তার কাছে।'

'কেন ফেলুদা?'

'এত বক বক করিসনে তো। যেটা বলছি মন দিয়ে শোন.... '

কৃত্রিম নিউরনে অনুরণন - পর্ব ২

সজির বাটিটা ফেলুন্দা হাতে নিয়ে বলল, "যারা যারা খুন হয়েছেন, তারা প্রত্যেকেই প্রত্যক্ষ বা পরোক্ষভাবে ওপেনসোর্স অ্যাস্ট্রিভিটির সাথে জড়িত।"

বললাম, "সেটা বুঝলে কী করে?"

- "ওরে বোকচন্দ, সেটা তো একটু গুগল আর ফিল্ডওয়ার্ক করলেই বোৰা যায়। তোকে যে কাজ দিয়েছিলাম তাৰ কী খবৰ?"

- "আমাৰ মনে হয় কাজটা অনেক জটিল, আমাৰ মাথায় কিছু আইডিয়া এসেছে, সেগুলো একটু প্ৰয়োগ কৰে দেখবখন।"

- "তা কৱিস, আৰ মাথায় বাখিস কাজটা খুবই দুট কৰতে হবে, সিধু জ্যাঠাৰ সাথেও একটু আলোচনা কৰতে এটা নিয়ে। দেখি উনি কিছু আলোকপাত কৰতে পাৱেন কিনা।"

এৱপৰ আৰ তেমন কোন কথা হল না, খাওয়া শেষ কৰে নিজেৰ রুমে ফিৰে গিয়ে কম্পিউটাৰ চালু কৰে ফাইলগুলো ওপেন কৰলাম।

পিক্সেল কাউন্টিং ও হিস্টোগ্রাম প্লটিং

যে কাজটা কৰব ভেবেছিলাম সেটা দিয়েই শুৰু কৰা উচিং বলে মনে হল। এখানে একটু বলে রাখা দৰকাৰ, আমি ছবিগুলো সব বাইনাৰি লেভেলে নিয়ে আসব `rgb2gray` ফাংশনেৰ মাধ্যমে, মানে কালো পিক্সেলগুলোৰ নিউমেৰিকাল ড্যালু ১ এবং সাদাগুলোৰ ০। এই দুই মান ছাড়া অন্য কোন মানেৰ পিক্সেল ছবিতে থাকবে না।

প্ৰথমে যেটা কৰা দৰকাৰ, আৰও কিছু হে঳াৰ ফাংশন লিখতে হবে যেটা দিয়ে আমি প্লটিং আৰ কাউন্টিংয়েৰ কাজ সহজেই কৰতে পাৱব।

```
import matplotlib.image as mpimg

# Plots histogram of a given digit
def plot_hist(digit, index):
    data = rgb2gray(mpimg.imread(load_digit(digit)[index])).ravel()
    plt.hist(data, alpha=0.5)
    plt.title("Histogram of " + str(digit) + " at index: " + str(index))
    plt.xlabel("Pixel Value")
    plt.ylabel("Count")
    plt.show()
```

এই ফাংশনেৰ সাহায্যে আমি যেকোন ডিজিটেৰ হিস্টোগ্রাম প্লট কৰতে পাৱব। এৱ আগে একটু হিস্টোগ্রাম সম্পর্কে বলি। হিস্টোগ্রাম হচ্ছে একটা ৱেঞ্জেৰ মধ্যে একটা ড্যালু কতবাৰ আছে তাৰ বাবে প্লট।

সহজভাবে বললে এভাবে ব্যাখ্যা করা যায়,

{

ধরি আমি একজন স্কুলশিক্ষক এবং আমি আমার 50 জন ছাত্রদের গণিত বিষয়ের নম্বরের ইস্টোগ্রাম প্লট করতে চাচ্ছি। তাহলে আমি প্রথমেই যে কাজটা করব সেটা হচ্ছে পরীক্ষায় সর্বোচ্চ নম্বর 100 কে আমি কতগুলো বেঞ্জে ভাগ করব। হিসাবের সুবিধার্থে আমি 100 কে 5 ভাগ করি তাহলে প্রতিভাগে বেঞ্জে হবে অনেকটা এরকম
 $0 - 20, 21 - 40, 41 - 56, 61 - 80, 81 - 100$

এই যে আমি কতগুলো বেঞ্জে ক্যাটাগরি করলাম এগুলোকে বলে এক একটা **bin**

এরপর আমি হিসাব করব, $0 - 20$ মার্ক পেয়েছে এমন কতজন ছাত্র আছে, $21 - 40$ মার্ক পেয়েছে এমন কতজন আছে এভাবে একটি ডেটা টেবিল তৈরি করব।

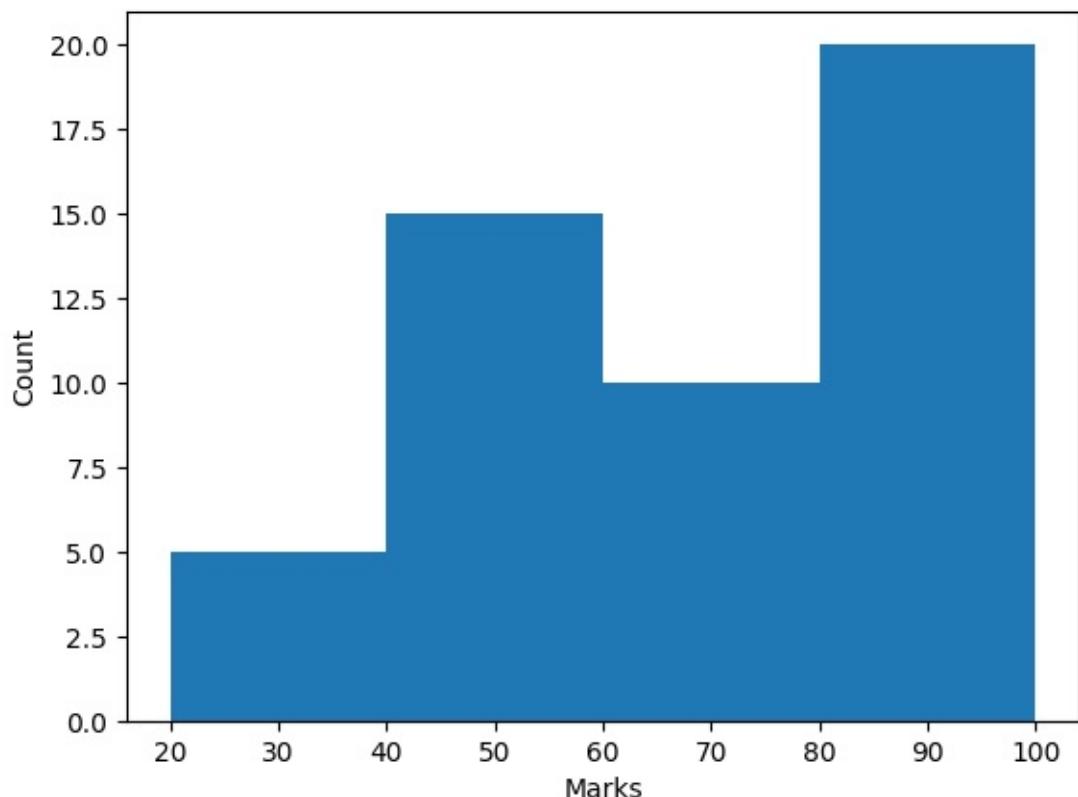
নিচের সাজানো ডেটাটেবিল পাইখনে অটোমেটিক তৈরি করা যায় শুধু `hist` ফাংশন কল করার মাধ্যমে।

প্রাপ্ত নম্বরের তালিকা

```
20, 20, 20, 20, 20,
40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40,
60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60,
80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80,
100, 100, 100, 100, 100
```

Bins of Mark (প্রাপ্ত নম্বর)	Count (কতজন পেয়েছে)
$0 - 20$	5
$21 - 40$	15
$41 - 60$	10
$61 - 80$	15
$81 - 100$	5

ପ୍ରାଣ ନମ୍ବର ଥିକେ ପ୍ରାଣ ହିସ୍ଟୋଗ୍ରାମ



1

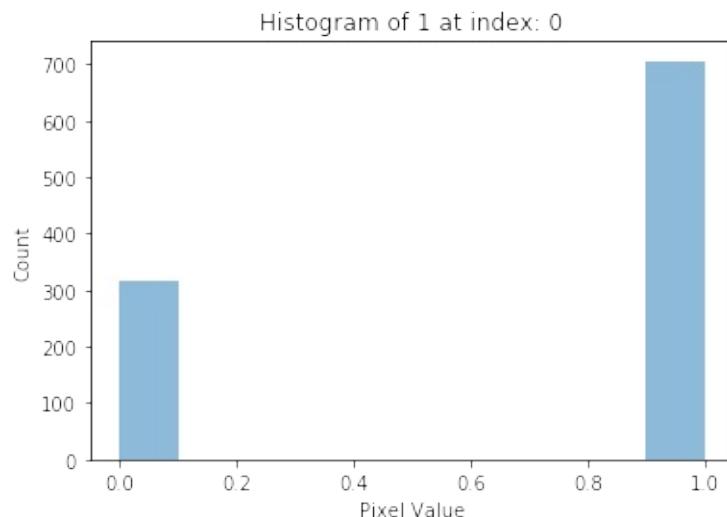
আমাদের ইমেজে আছেই দুইরকমের মান, একটি হল ০ আরেকটি হল ১। সেক্ষেত্রে হিস্টোগ্রাম আসবে দুইটা বার কিন্তু তাদের হাইট কৃত হবে সেটা নির্ভর করবে ০ বা ১ কৃতবার আছে তার উপর।

একটা ঈমেজের হিস্টোগ্রাম বের করা দেখা যাক আমার তৈরিকৃত ফাংশন দিয়ে।

আমি এখন '১' এর ঘৃতগুলো ছবি আছে তাদের লিঙ্গের পথম ছবিটার টিক্টোরাম প্লট কুরু নিচে কেড়ে দিয়ে।

```
# Plotting histogram of digit 1 from index 0
plot_hist(1, 0)
```

আউটপুট



এখন দেখতে হবে প্রতিটা ডিজিটের ইস্টোগ্রাম কতটা আলাদা হয়, কারণ তাহলে আমি একটা প্রেসহোল্ডিং বসাতে পারব যে \circ পিক্সেল ভ্যালু এর সংখ্যা বা \rangle পিক্সেল ভ্যালু এর সংখ্যা এতগুলো হলে সেটা $\rangle, \circ, \circ, \circ, \circ$ ইত্যাদি ডিজিট হবে।

৩ ২ এর মাঝে তুলনা

দুইটা ডিজিট যদি আমি চিনতে চাই তাহলে তাদের মধ্যে এমন একটা সাধারণ বৈশিষ্ট্য থাকবে যেটার মাধ্যমে আমি আলাদা করতে পারব। যেমন হতে পারে কতগুলো কালো/সাদা পিক্সেল প্রতি ইমেজে আছে। যদি আমি দেখি যে \rangle এর ক্ষেত্রে কালো পিক্সেলের সংখ্যা \approx এর তুলনায় ৫০ টা বেশি থাকে তাহলে আমি বলতে দুইটা ইমেজ তুলনা করে বলে দিতে পারব কোনটা \rangle এবং কোনটা \approx ।

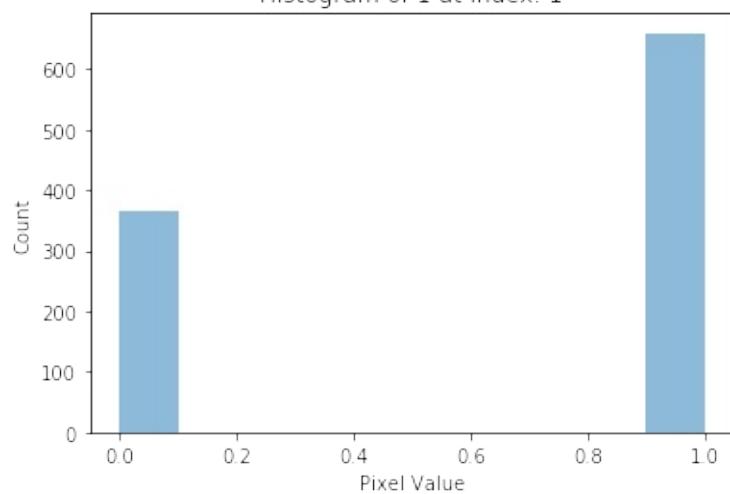
ইস্টোগ্রাম প্রটিংয়ের মাধ্যমে তুলনা

সংখ্যাগত তুলনায় যাওয়ার আগে ডিজুয়াল কম্প্যারিজনের বিষয়টা মনে ধরল। তাই সংখ্যা ১ এর কয়েকটা স্যাম্পলের ইস্টোগ্রাম প্লট করে দেখব কোন প্যাটার্ন পাওয়া যায় কিনা।

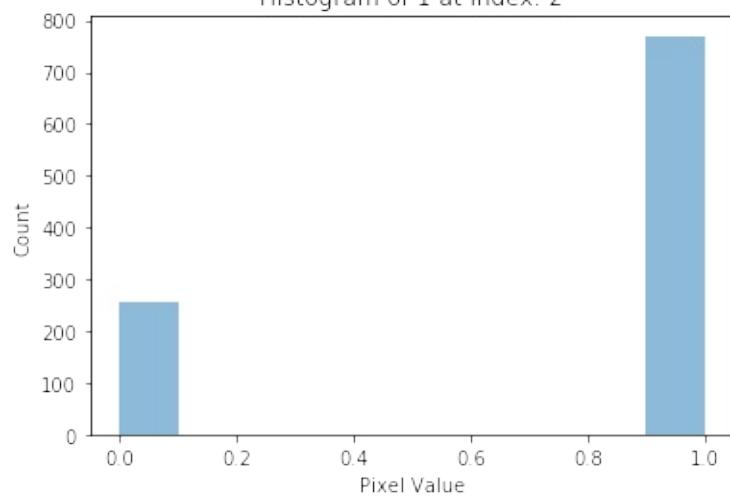
```
# Plotting histogram of five instances of digit 1
for i in range(5):
    plot_hist(1, i)
```

চারটা ইস্টোগ্রাম প্লট পেলাম,

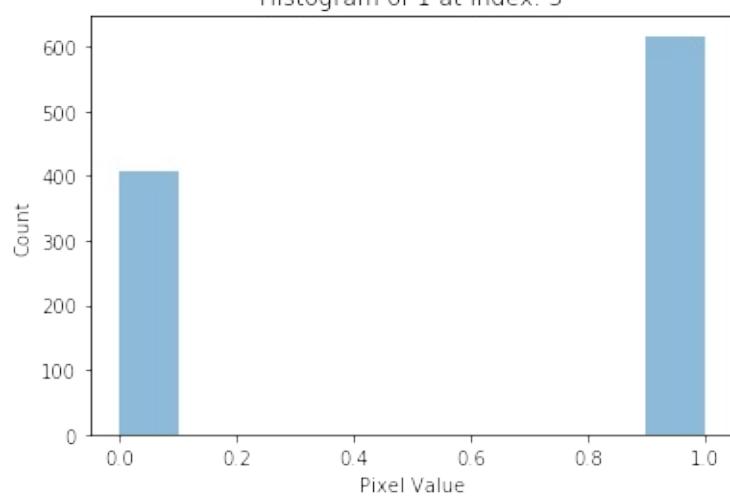
Histogram of 1 at index: 1

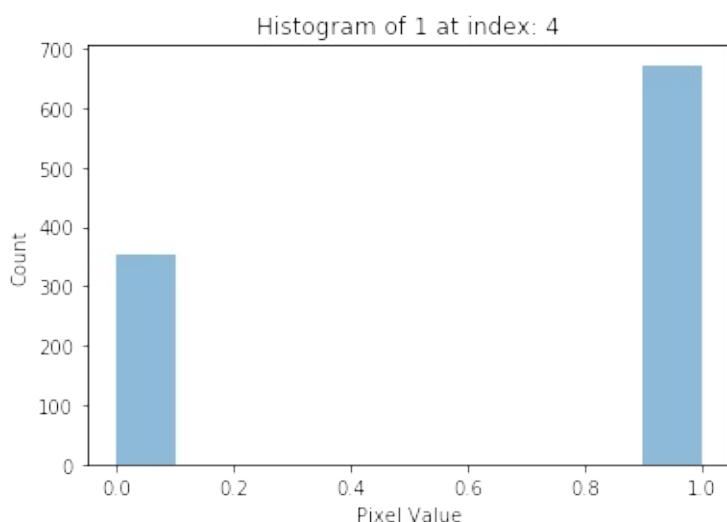


Histogram of 1 at index: 2



Histogram of 1 at index: 3





এখনে একই সংখ্যার বিডিও স্যাম্পলের হিস্টোগ্রাম প্লট করলাম। আপানুরূপ ফলাফল পাচ্ছি না কারণ সাদা ও কালোর পিঙ্গেল সংখ্যার কোন নির্দিষ্ট মান নেই!

নিউমেরিকাল ভ্যালু দিয়ে তুলনা

এই তুলনা করার আগে আমার দেখতে হবে আমি যে ভিত্তিতে আগাছি সেটা কাজ করবে কিনা। তাই আমার অনেকগুলো স্যাম্পল নিয়ে আমার ধারণা প্রয়োগ করে দেখতে হবে আমি কতটা ঠিক বা ভুল। আমি ১ ও ২ এর থেকে প্রতি ডিজিটের ১০০ টি করে ইমেজ নিয়ে গড় করে দেখব ১ ও ২ সংখ্যায় কালো ও সাদা পিঙ্গেল কতগুলো আছে। যদি একটা বিশাল পার্থক্য পাই তাহলে খুব সহজেই আমি এমন একটি সিস্টেম তৈরি করতে পারব যেটা ১ ও ২ চিনতে পারে।

আর দেরি না করে ঝটপট শুরু করলাম,

```

# Returns the pixel count by category
def pixel_count(digit, index):
    # Reading an image, squishing the multidimensional array into single dimension
    data = rgb2gray(mpimg.imread(load_digit(digit)[index])).ravel()
    ones = [i for i in data if i == 1]
    zeros = [i for i in data if i == 0]
    value_dict = {
        "white_count" : len(ones),
        "black_count" : len(zeros)
    }
    return value_dict

# Comparison between two digits [applying basic statistics]
one_black_pixels = []
one_white_pixels = []

two_black_pixels = []
two_white_pixels = []

for i in range(100):
    one_black_pixels.append(pixel_count(1, i)['black_count'])
    one_white_pixels.append(pixel_count(1, i)['white_count'])

    two_black_pixels.append(pixel_count(2, i)['black_count'])
    two_white_pixels.append(pixel_count(2, i)['white_count'])

# Calculating the mean
one_black_pixel_mean = np.mean(one_black_pixels)
one_white_pixel_mean = np.mean(one_white_pixels)

two_black_pixel_mean = np.mean(two_black_pixels)
two_white_pixel_mean = np.mean(two_white_pixels)

print("1 [white] : {0}\n1 [black] : {1}".format(one_white_pixel_mean,
                                                one_black_pixel_mean
                                               ))
print(" -- ")
print("2 [white] : {0}\n2 [black] : {1}".format(two_white_pixel_mean,
                                                two_black_pixel_mean
                                               ))

```

ଆଡ଼ଟପ୍ଟ

```

1 [white] : 646.16
1 [black] : 377.84
--
2 [white] : 726.47
2 [black] : 297.53

```

অর্থাং

সংখ্যা ১ এর ১০০ টি ইমেজ নিয়ে তাদের সাদা ও কালো পিক্সেলের সংখ্যা গড় করলে আমরা পাই যথাক্রমে 646.16 এবং 377.84

সংখ্যা ২ এর ক্ষেত্রে সেটা 726.47 এবং 297.53 এতটি।

একটা পার্থক্য লক্ষণীয়। কিন্তু আমাকে গড় না করে আলাদাভাবে তাদের কাউন্ট দেখতে হবে। গড় সবসময় এক্সট্রিম মানের দিকে ধাবিত হয়।

১ ও ২ এর পিক্সেল কাউন্ট এর পার্থক্যের বিশ্লেষণ

আমি সংখ্যা ১ ও ২ থেকে প্রতি সংখ্যার ৫০ টি করে স্যাম্পল নেব। তারপর তাদের সাদা পিক্সেল কাউন্টের পার্থক্য একটি অ্যারেতে রাখব। আমি আসলে দেখতে চাচ্ছি ১ ও ২ এর মধ্যে পিক্সেল কাউন্ট কতটা পরিবর্তিত হয়। যদি মিনিমাম পিক্সেল সংখ্যা একদম কম হয় তাহলে আমি তাদেরকে শুধু পিক্সেল কাউন্টের মাধ্যমে আলাদা করতে পারব না।

```

# Finding out difference between white count between 1 and 2
values = []
for i in range(50):
    values.append(abs(pixel_count(1, i)['white_count'] - pixel_count(2, i)['white_count']))
    ## If checking is needed
    #print(values[i])

# Maximum and minimum of the collected difference
print ("Minimum: {}".format(np.min(values)))
print ("Maximum: {}".format(np.max(values)))

```

আউটপুট:

```

Minimum: 6
Maximum: 335

```

নাহ, মিনিমাম একদমই কম! যদি ৫০-১০০ হত তাহলে হ্যাত একটা ফ্রেজহেল্ড দিতে পারতাম, কিন্তু এখন সেটা সম্ভব হবে না। আমি আবার এখানে `absolute` মান ব্যবহার করেছি। এর মানে হল, এমনও হতে পারে ১ এ সাদা পিক্সেলের সংখ্যা ২ এর চেয়ে বেশি অথবা কম! তার মানে পিক্সেল কাউন্ট পদ্ধতিতে আদৌ বোঝার কোন উপায় নেই!

ଦିଶେହାରା ହୟେ ଗେଲାମ । ଏମନ ସମୟ ଫେଲୁଦା ଘରେ ପ୍ରବେଶ କରଲ ।

-'ସିଧୁଜ୍ୟଠାକେ ଫୋନ କରେ ପେଲାମ, ଚଳ ଦେଖା କରେ ଆସି ।'

-'ଫେଲୁଦା, ଆମି ଯେ ଆଇଡ଼ିଆ ବେର କରେଛି ସେଟା କାଜ କରଛେ ନା ।'

-'ତୋର ଆଇଡ଼ିଆଟା କୀ ଛିଲ ?'

-'ସାଦା ଆର କାଲୋ ପିଞ୍ଜେଲ କାଉଟ୍ କରେ ସଂଖ୍ୟା ନିର୍ଧାରଣ !'

କିଛୁକ୍ଷଣ ଆଞ୍ଚୁଲ ମଟକିଯେ ଫେଲୁଦା ବଲଲ,

-'ଟୋଟାଲ ପିଞ୍ଜେଲ କାଉଟ୍ ଯଦି ପ୍ରାୟ ଏକଇ ଆସେ ସେକ୍ଷେତ୍ରେ ତୁଇ ପଜିଶନ ଭେଦେ ପିଞ୍ଜେଲ ଡେନସିଟି ମେଜାର କରତେ ପାରିସ । ଯେମନ ଆମରା ୪ ଲିଖଲେ ମାଝଥାନେର ଦିକେ ଯତଟା କାଲୋ ପିଞ୍ଜେଲ ପାଓଯା ଯାବେ ୦ ତେ ସେଇ ତୁଳନାୟ ଅନେକ କମ ପାଓଯା ଯାବେ ଏବଂ କୋନ କ୍ଷେତ୍ରେ ଯାବେଇ ନା ।'

-'ବୁଝଲାମ ନା, ଆରେକଟୁ ପରିଷାର କରେ ବଲ ତୋ ?'

-' ୫ × ୫ ସାଇଜେର ଏକଟା **2D Square Magnifying** ମାସେର କଥା ଚିତ୍ରା କର, ଏଟା ଆମି ଇମେଜ ମ୍ୟାଟ୍ରିଙ୍ଗେର ଏକଟା ଅଂଶେ ପ୍ରଜେଷ୍ଟ କରଲେ ସେଖାନକାର ପିଞ୍ଜେଲ ଗୁଲୋ ଦେଖିତେ ପାବ, ତାଇ ନା ?'

'ହଁଁ, ତାହଲେ ?'

-'ତାହଲେ ମ୍ୟାଗନିଫାଇୟିଂ ଫ୍ଲାସ ଟା ଆମି ଯଦି ଆରଓ ଡାନେ ୫ ଘର ବା ଛବିର କ୍ଷେତ୍ରେ ପିଞ୍ଜେଲ ସରାଇ, ତାହଲେ କୀ ଆମି ଆଗେର ପିଞ୍ଜେଲଗୁଲୋ ଦେଖିତେ ପାବ ?'

-'ନା !', ପରକ୍ଷଣେଇ ଫେଲୁଦାର କଥା ବୁଝିତେ ପାରାତେ ବଲଲାମ, 'ଆଜ୍ଞା ତାରମାନେ ଆମି ଏକଟା ଆବିର୍ତ୍ତରୀବି ସାଇଜେର ମ୍ୟାଟ୍ରିଙ୍ଗ କେ ଏତାବେ **2D** ଫ୍ଲେନେ ଇଟାରେଟ କରେ ପଜିଶନଓୟାଇଜ ପିଞ୍ଜେଲ ଡେନସିଟି ମାପବ ? !'

-'ହଁଁ, ସେଟା ପରେ କରିସ, ଆପାତତ ଆର କଥା ନୟ ଏକ୍ଷୁଣି ବେର ହତେ ହବେ ଆମାଦେର ।'

ମନେ ମନେ ଫେଲୁଦାର ତାରିଫ ନା କରେ ପାରଲାମ ନା, ଓର ଅନେକଗୁଲୋ ଗୁଣେର ମଧ୍ୟେ ଏକଟା ହଳ; ନତୁନ କୋନ ସମସ୍ୟା ହାତେ ଏଲୋଇ ଓ ସେଟାର ଉପର ଟାଙ୍କି କରା ଶୁରୁ କରେ ଦେଯ ଏବଂ ଯତ ଦ୍ରୁତ ସଭବ ସେଟା ସମ୍ପର୍କେ ଆପଡେଟେଡ ହୋଯାର ଚେଷ୍ଟା କରେ । ଆମି ଯେ କରେ ଶିଖିବ ସେଟାଇ ବୁଝିତେ ପାରାଇଁ ନା ।

ଜାନ୍ଯୁଆରିର ମାଝମାଝି, ଶୀତ ମୋଟେ ପଡ଼ା ଶୁରୁ କରେଛେ, ତାତେଇ ବାଇରେ ବେଶ ଠାଣା, ଆର ବିକେଲେର ଦିକେ ତୋ ଏଲୋମେଲୋ ଦମକା ହାଓଯାଯ ବେରନୋଇ ମୁଣ୍ଡିଲ । ସେବ କିଛୁର ତୋଯାଙ୍କା ନା କରେ ଏକଟା ପୁଲଓଭାର ଚାପିଯେ ବେର ହୟେ ଗେଲାମ ଫେଲୁଦାର ସାଥେ ।

-'ଫେଲୁ ମିତିର, ଏତଦିନ ପର ହଠାଏ ଶ୍ମରଣ କରଲେ ଯେ? ପ୍ରାୟ ଖୁବ ବେଡ଼େ ଗେଛେ ମନେ ହୟ?', ସୋଫାଯ ବସେ ବଲଲେନ ସିଧୁଜ୍ୟଠା ।

-'ଆପନାର ଆଶୀର୍ବାଦେ ତା କିଛୁଟା ବେଡ଼େଛେ ବୈକି!'

-'ତୁମି ନିଶ୍ଚଯିତ୍ବ ଅଧ୍ୟାପକ ହତ୍ୟାକାଣ୍ଡେ ବ୍ୟାପାରେ ଆମାର ସାଥେ ଦେଖା କରତେ ଏମେହ ?'

একটা চওড়া হাসি দিয়ে ফেলুদা বলল, 'কীভাবে বুঝলেন?'

-'কারণ, তুমি জান আমি ইটারেস্টিং ঘটনা পড়তে ও সংগ্রহ করতে ভালবাসি, শুধু শুধু একজন মধ্যাবিত্ত অধ্যাপককে কেন খুন করা হল সেটা বেশ রহস্যজনক ব্যাপার। ভদ্রলোক যেহেতু ওপেনসোর্স অ্যাস্টিভিটির সাথে জড়িত ছিল, তোমার ধারণা সেকারণেই হ্যাত উনি খুন হন। আর আমি নিজেও ওপেনসোর্স কমিউনিটির একজন অ্যাস্টিভ সদস্য ছিলাম। তাই দুইয়ে দুইয়ে চার করতে কোন সমস্যা হ্য নি।'

-'আপনি গোয়েন্দাগিরি করলে আমাদের মত প্রাইভেট ইনভেস্টিগেটরদের পথে বসতে হত!', পরিবেশ হাস্তা করে ফেলুদা বলল।

-'হাহা, আমি অনেক কিছু করলে অনেকেরই পথে বসতে হত, তাই কিছুই করিনি! শুধু মাইন্ড ওপেন রেখে যতটা সম্ভব জ্ঞানার্জন করা যায় শুধু সেই পথেই হাঁচছি! তা বল, কী জানতে চাও?'

-'অধ্যাপকের পাশাপাশি শশাঙ্ক মুখাজী নামের একজন ভদ্রলোক খুন হন, ওনার পেশা জানা যায় নি। এমনকি প্রতিবেশীরাও জানে না এ ব্যাপারে কিছু বলতে পারছে না। আমি ধারণা করছি এই দুই ভদ্রলোকই কোনভাবে সম্পর্কিত। অভিযাসলি এটাও একটা কারণ যে তাঁদেরকে একইভাবে হত্যা করা হ্য। আপনি শশাঙ্কবাবু সম্পর্কে কিছু জানেন কিনা সেটা জিজ্ঞাসা করতে আসলাম'

কিছুক্ষণ সবাই চুপচাপ, বেয়ারা এসে চা, চানাচুর আর বিস্কুট দিয়ে গেল। কিছু করার ছিল না দেখে আমি একটা বিস্কুট নিয়ে খেতে শুরু করলাম। সিধুজ্যাঠা বললেন, 'নাহ, নামটা একদম মনে পড়ছে না, আচ্ছা এক মিনিট।', বলেই ঘরের ডেতরে চলে গেলেন।

ঠিক প্রায় তৎক্ষণাত্তে একটা বই হাতে প্রবেশ করলেন,

'এটাকে বলতে পারো ওপেনসোর্স কন্ট্রিবিউটর ডিরেষ্টরি, এখন এইসব জিনিস আমার কাছে আছে জানলে পুলিশে ধরে নিয়ে যাবে। যদি শশাঙ্কবাবু সত্যিই ওপেনসোর্স অ্যাস্টিভিটির সাথে জড়িত থাকেন তাহলে অবশ্যই তার বায়ো এখানে থাকবে।'

বলে বইটা টেবিলে রেখে পাতা ওল্টাতে শুরু করলেন। আমি বেশ আগ্রহের সাথে দেখতে থাকলাম। টেকনোলজিক্যাল অ্যাডভান্সমেন্টের অন্যতম হাতিয়ার ওপেনসোর্স কন্ট্রিবিউশন। এত এত কোম্পানি দাঁড়িয়ে আছে ওপেনসোর্স টুলস কে পুঁজি করে। হ্যাত কন্ট্রিবিউশন এখন নিষিদ্ধ কিন্তু রিসার্চ থেকে শুরু করে সবকাজের ডিতি ছিল এই ওপেন সফটওয়্যার স্ট্যাক।

ডিরেষ্টরি ঘাঁটতে ঘাঁটতে সিধুজ্যাঠা বললেন, 'নাহ এখানে তো ওঁ নামে কাউকে খুঁজে পাচ্ছি না!' এমন সময় আমি জোরে বলে উঠলাম, 'সিধু জ্যাঠা থামুন!'

সিধুজ্যাঠা চমকে আমার দিকে তাকাতে ফেলুদা ডিরেষ্টরি থেকে চোখ না সরিয়ে বলল, 'মানসবাবুও তাহলে ওপেনসোর্স কন্ট্রিবিউটর ছিলেন!'

-'ও আচ্ছা উনি? উনি তো তেমন কন্ট্রিবিউট করতেন না, তখন একটা ট্রেন্ড ছিল এবং কন্ট্রিবিউটরদের অন্য চোখে দেখা হত। উনি বোধহয় হজুগে নিজে নাম এন্ট্রি করেছিলেন। উল্লেখযোগ্য কোন কন্ট্রিবিউশন তার ঝুলিতে নেই। তোমরা চেনো কীভাবে এঁকে?' বললেন সিধু জ্যাঠা।

-'সে অনেক কাহিনী সিধু জ্যাঠা, আপনাকে অন্য সময় বলব। হতে পারে এটা কোইনিডেম, শশাঙ্ক মুখাজীর মৃত্যুর কারণ হ্যাত অন্য কিছু। যাই হোক, একটা সিস্টেম বিন্দ করতে চাচ্ছি সেটা হাতে লেখা বাংলা ডিজিট রিকগনাইজ করতে পারে। তোপসেই মূল কাজ করছে কিন্তু এ ব্যাপারে আপনার মতামত শুনলে ভাল হত!', বলল ফেলুদা।

-'আচ্ছা, তোমরা নিশ্চয়ই নলেজ বেজড মেথডে আগাছ? সেটা খুব একটা ভাল কাজ করবে না, তোমরা বরং স্ট্যাটিস্টিক্যাল ও প্রব্যাবিলিস্টিক অ্যানালাইসিস কর। আমি একটা বই দিয়ে দিচ্ছি, পড়লে বেশ কিছু জিনিস পরিষ্কার হবে। আর কোন কিছু ইম্প্রিমেটেশনে সমস্যা হলে আমাকে বলতে পার। তবে ঘুণাক্ষরেও কেউ যেন টের না পায় তোমরা এ জিনিস ডেভেলপ করছ!'

বলে সিধুজ্যাঠা আবারও ঘরে চলে গেলেন ও কিছুক্ষণ পরেই চলে আসলেন, তার হাতে একটা ছোট বই। বইটা ফেলুদার হাতে দিয়ে বললেন, 'বইটা বেশ ভাল, তবে আবারও একই কথা। খুব সারধানে কাজ করবে। বেস্ট অফ লাক!'

প্রব্যাবিলিটি কলেজে থাকতে পড়েছিলাম, মনে নেই কি পড়েছিলাম। সম্ভবত বুঝিনি, যাই হোক সিধু জ্যাঠার দেওয়া বইটার নাম, [A First Course in Probability](#)।

বাড়িতে ফিরেই আমাকে ফেলুদা বলল, 'আমি রাতে আর কিছু খাব না, ডিস্টাৰ্ব করবি না একদম। যে কাজ দিয়েছি গুডবয়ের মত করে রাখিস, তোর সাথে এ নিয়ে কাল আমার কথা হবে। বুঝেছিস?'

আমি মাথা নেড়ে নিজের ঘরে চলে গেলাম। কাজটা করতে বেশ আলসেমি লাগছিল। তাই মনশ্বির করলাম একটু বাইরে থেকে বেড়িয়ে আসি।

রাতের বেলা ফিরে শঁকরকে ফেলুদার কথা জিজ্ঞেস করতে বলল, 'ফেলুবাবু তো ঘর থেকে বের হননি, তোমার খাবার কি দিয়ে দেব?'

আমি মাথা নেড়ে চিন্তা করতে থাকলাম, ফেলুদা কি আজকের মধ্যে বইটা শেষ করবে নাকি? নাকি ও রহস্যের জট ছাড়াচ্ছে? আর বেশিক্ষণ চিন্তা করলে হ্যাত আমার নিজের মাথায়ই জট লেগে যেত তাই আমি চিন্তা করা বাদ দিয়ে খেয়ে ঘুমিয়ে গেলাম।

সকালে উঠেই বৈঠকখানায় ফেলুদাকে পেয়ে গেলে বললাম, 'কিছু কুলকিনারা করতে পারলে?'

ফেলুদা সে কথার জবাব না দিয়ে হাতের কয়েন দেখিয়ে বলল, "এই কয়েনটা আমি যদি টস করি তাহলে Head উঠবে নাকি Tail ?"

বললাম, 'এত খুব সহজ, ৫০%-৫০% চাঞ্চ!'

-'তুই ৫০-৫০ কীভাবে হিসাব করলি? আর এক্যাটলি বলতে পারবি হেড হবে নাকি টেইল?'

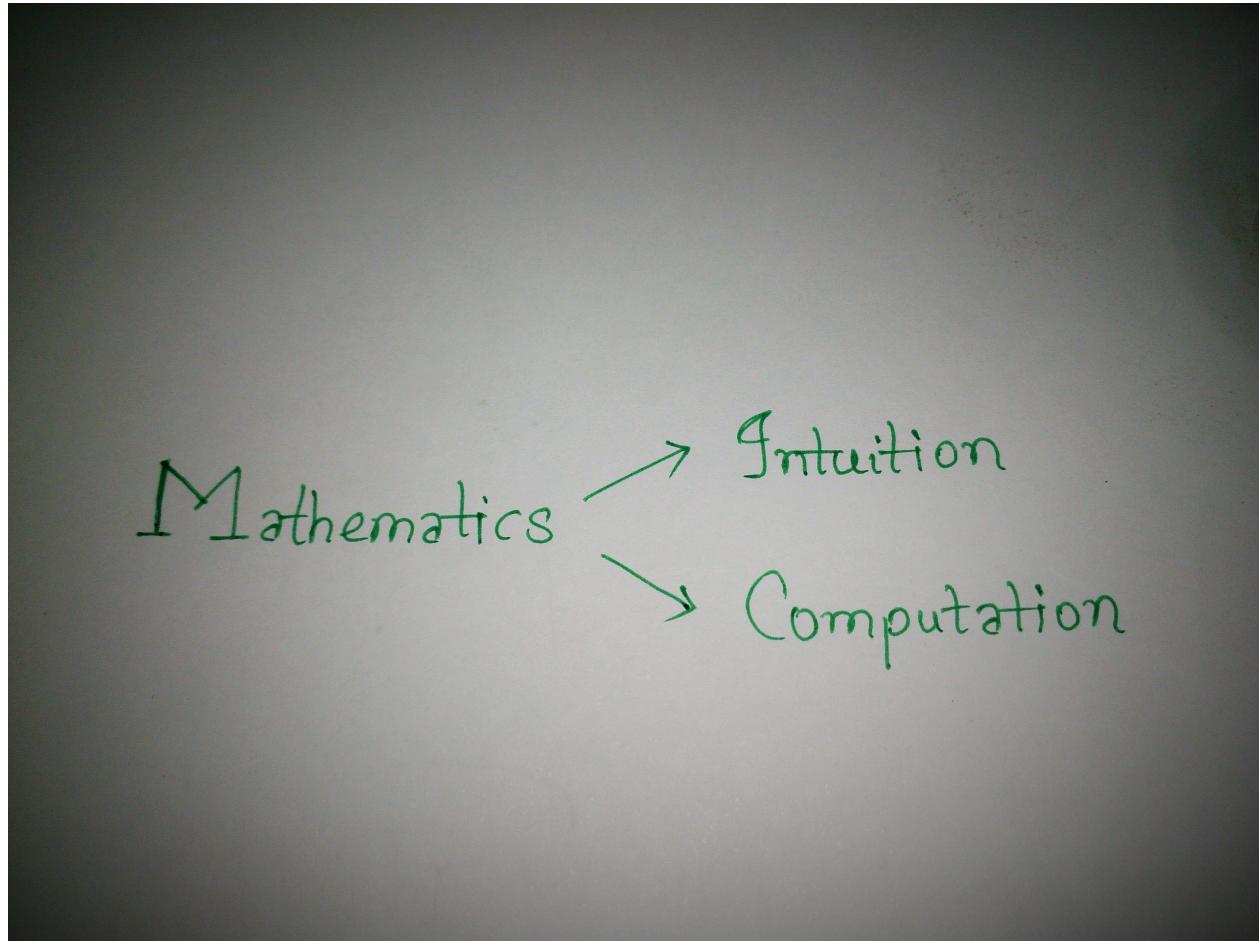
-'ইনটুইশন দিয়ে! এক্যাটলি কি হবে সেটা বলতে পারব না, সেতো খুব কঠিন কাজ!'

আচ্ছা, এবার আরেকটা প্রশ্ন করি,

-'ধৰ, আমি কয়েনটা একটু বাঁকিয়ে ফেললাম। এরপর তোকে বললাম, পরপর তিন টসে দুইটা হেড ওঠার সম্ভাবনা কত?'

কিছুক্ষণ মাথা চুলকে বললাম, 'নাহ ফেলুদা, পারছি না!'

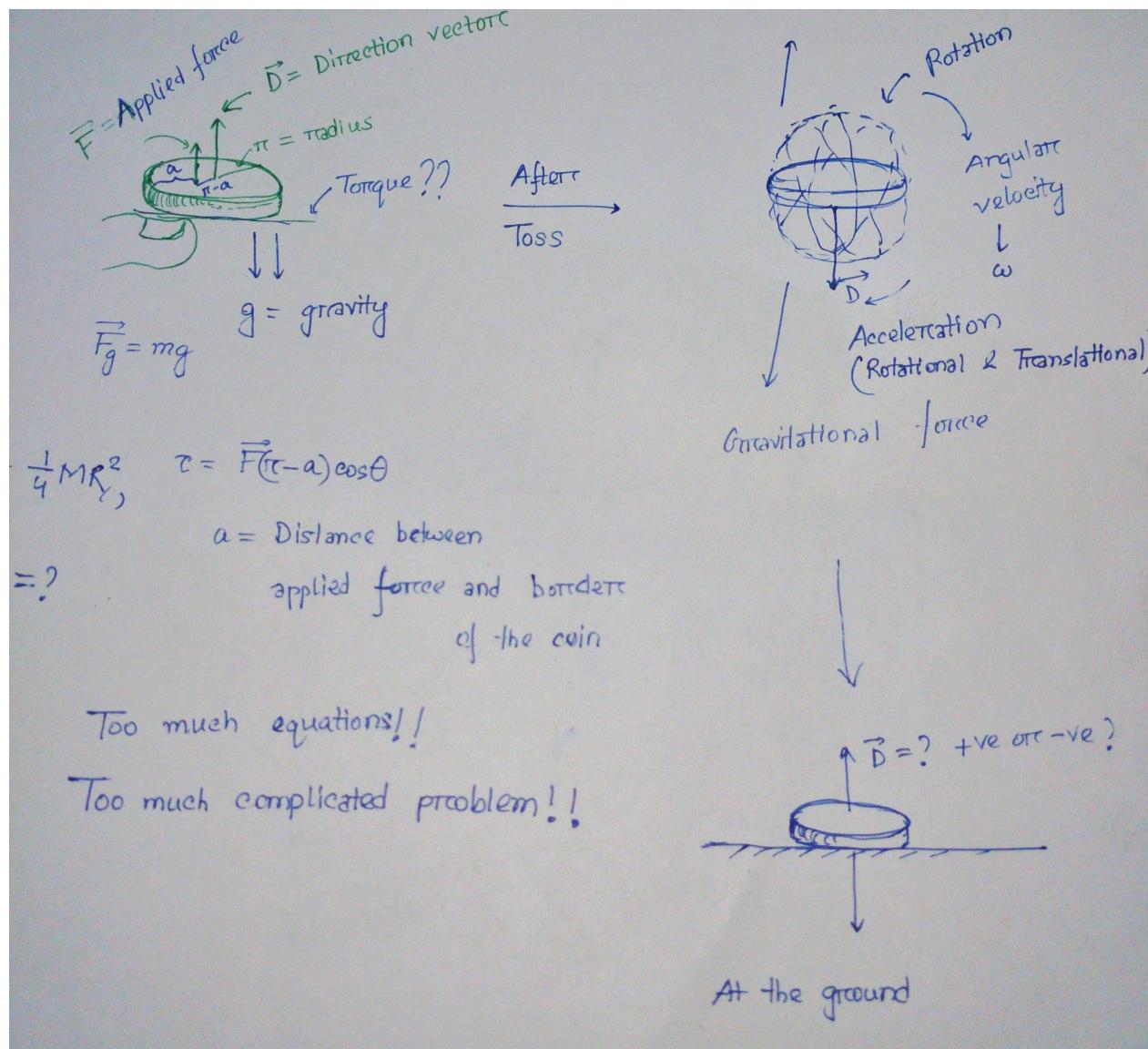
-"পড়ালেখা কি করিস কিছু? কি ম্যাথ শিখায় তোদের স্কুলে? তোকে ধমক দিয়ে আসলে লাভ নেই, শিক্ষাব্যবস্থাই এরকম। গণিতে আমরা ইন্ট্রুইশন থেকে কম্পিউটেশন বেশি করি আর সে কারণেই ম্যাথ বোরিং লাগে। কম্পিউটেশনের জন্য আমাদের তো কম্পিউটার আছেই। সবথেকে বেশি জরুরি ইন্ট্রুইশন।"



"আচ্ছা এটা বলতে পারবি কেন এক্স্যাট্টলি বলাটা টাফ যে n^{th} টস Head হবে নাকি Tail?"

মাথা নাড়তেই ফেলুন্দা বলল, "নাহ, তোকে দিয়ে কিছু হবে না। এখন দেখ, " ধমক দিয়ে ফেলুন্দা একটা কাগজে লেখা শুরু করল সাথে বলতে লাগল,

"ধর এটা একটা কয়েন যেটা আমি টস করব, ফিজিক্সে একে যদি আমি ইকুয়েশন দিয়ে মডেল করতে চাই তাহলে ব্যাপারটা হবে এরকম"



"দেখেছিস কতটা জটিল হয়ে যাচ্ছে? এসব প্রবলেমকে ম্যাথেমেটিক্যালি হ্যান্ডেল করতেও জটিল মডেল সিম্পল করতে আমরা প্রয্যাবিলিটির সাহায্য নেই। আমি যদিও কোনভাবে ফিজিক্স খাটিয়ে মডেল তৈরি করি তবুও তুই অতগুলো মান কখনোই ওঁ সময়ে মাপতে পারবি না। আচ্ছা তুই যে ৫০-৫০% চাঞ্চ বললি, সেটা কী হাতে কলমে দেখাতে পারবি?"

নাহ, সেটাও আমার দ্বারা সম্ভব না, তাই চুপ করে থাকাই বুদ্ধিমানের কাজ মনে হল।

ফেলুন্দা ওর ল্যাপটপ বের করে পাইথনে একটা কোড লিখল,

```

import numpy as np
import matplotlib.pyplot as plt

# Expressing Head|Tail in terms of 1|0
TAIL = 0
HEAD = 1

SMALL_SAMPLE_COUNT = 10
LARGE_SAMPLE_COUNT = 5000

## Taking 10 samples & 5000 samples for plotting histogram
coin_toss_10_samples = np.random.choice([TAIL, HEAD], SMALL_SAMPLE_COUNT)
coin_toss_5000_samples = np.random.choice([TAIL, HEAD], LARGE_SAMPLE_COUNT)

data = [coin_toss_10_samples, coin_toss_5000_samples]
titles = ['10 Coin Toss Result', '5000 Coin Toss result']

f, a = plt.subplots(2, 1)

a = a.ravel()

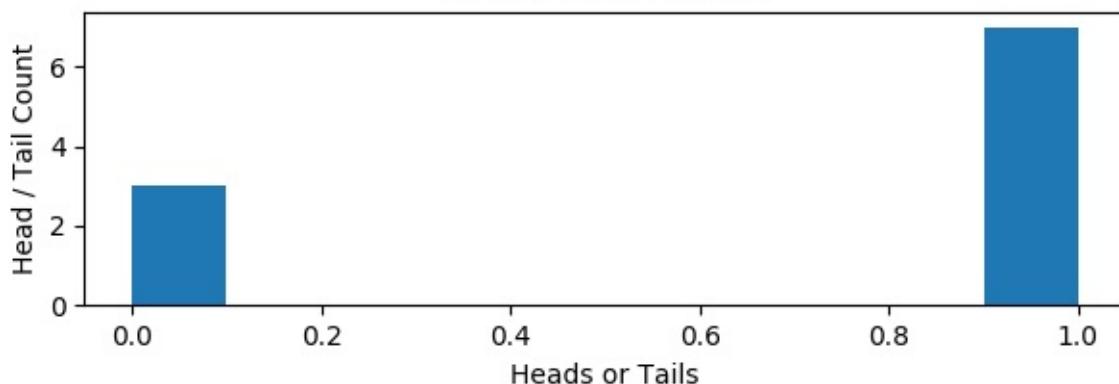
for index, axis in enumerate(a):
    axis.hist(data[index])
    axis.set_title(titles[index])
    axis.set_xlabel("Heads or Tails")
    axis.set_ylabel("Head / Tail Count")

plt.tight_layout()
plt.show()

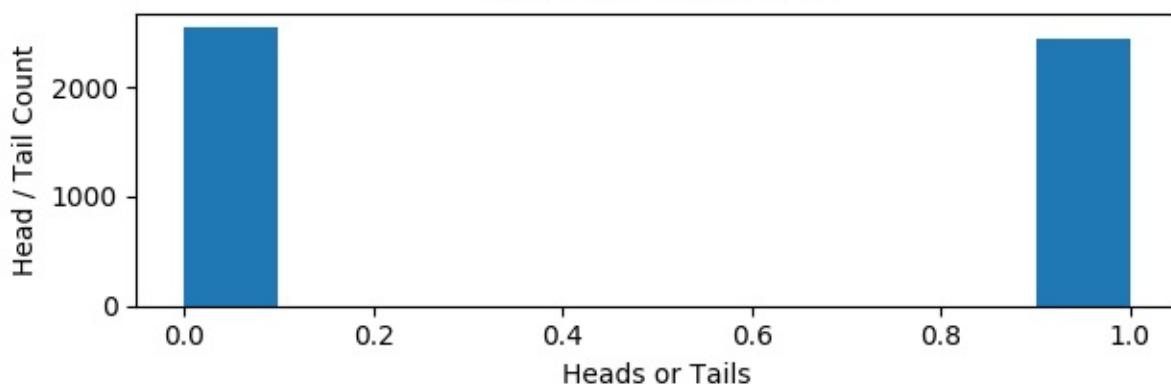
```

ଯାତ କରାତେ ଏଟା ଆସଲ,

10 Coin Toss Result



5000 Coin Toss result



"প্রথমে আমি যখন ১০ বার টস করলাম, দেখ যতটা না Tail এসেছে তার তুলনায় Head এর সংখ্যা বেশি (দ্বিগুণেরও বেশি!)। কিন্তু আমি যখন ৫০০০ বার টস করলাম তখন দেখ পার্থক্য কতটা কমে আসছে? এখন কিন্তু বলা যেতেই পারে $Head : Tail = 50 : 50 (approx)$ ।

Central Limit Theorem বলে, স্যাম্পলের সংখ্যা যত বেশি হবে.."

কৃত্রিম নিউরনে অনুরণন: পর্ব - ৩

"...রংয়াঙ্গম ড্যারিয়েবলের মানগুলো তত নরমালি ডিস্ট্রিবিউট হবে! এখান থেকেই নরমাল ডিস্ট্রিবিউশন বা গাউসিয়ান (Gaussian) ডিস্ট্রিবিউশনের উৎপত্তি কিন্তু অনেকে মনে করে নরমাল ডিস্ট্রিবিউশনের ফলাফল সেন্ট্রাল লিমিট থিওরেম।"

মাথামুণ্ডু কিছুই বুঝলাম না, ফেলুন্দা একনজর দেখে একটা চারমিনার ধরিয়ে বলা শুরু করল,

-"তোর ক্লাসে খুব লম্বা কয়জন ক্লাসমেট আছে?"

-"উম, বেশি নয় দুই একজন হবে।"

-"খুব খাটো?"

-"তাও বেশি নয়, দুই তিনজন।"

-"বাকি সবাই মাঝারি?"

-"হ্যাঁ" বলে চিন্তা করতে থাকলাম ফেলুন্দা হঠাৎ আমার ক্লাসমেটদের হাইট নিয়ে এত চিন্তিত কেন। পরক্ষণেই ও বলল,

-"এটাই হল নরমাল ডিস্ট্রিবিউশন!"

-"মানে?"

-"মানে হল, তোদের সবার হাইট নিয়ে আমি যদি গড় করি তাহলে দেখা যাবে গড়ের মান মাঝারি হাইটের আশেপাশেই আছে। তোদের ক্লাসে যদি বেশি হাইটের ছাত্র বেশি থাকত তাহলে গড়ের মান অবশ্যই বেশি আসত। শুধু তোর ক্লাসে না, তোর আশেপাশে একটু নজর দিলেই দেখতে পারবি সবকিছুই নরমালি ডিস্ট্রিবিউটেড। তবে আমি শুধু তোর ক্লাস বিবেচনা করলে সেটা কিছুটা ইউনিফর্ম হবে। কিন্তু যখন আমি আরও কয়েকটি ক্লাসের সবার হাইট মাপব তখন সেই ডিস্ট্রিবিউশনটা নরমাল ডিস্ট্রিবিউশনের আকার ধারণ করবে। গাউসিয়ানের দুইটা প্যারামিটার হল এক্সপেক্টেশন বা গড় এবং ড্যারিয়েন। আমি তোদের সব ক্লাসের সবার গড় আর ডারিয়েন পেলে আমি বলে দিতে পারব ঠিক কত শতাংশ ছাত্রের উচ্চতা ঠিক কতটুকু হতে পারে।"

ছাইদানিতে ছাই ঘেড়ে ফেলুন্দা বলতে থাকল,

"এখন যদি তুই বাস্কেটবল প্রেয়ারদের কথা চিন্তা করিস, তাহলে দেখবি তারা হাইটে কমবেশি একই। একে আমরা বলব ইউনিফর্ম ডিস্ট্রিবিউশন।

ম্যাথমেটিক্স যতটা আকাশ কুসুম ভাবিস অতটা কিন্তু না, কারণ কার্ল ফ্রিডরিখ গাউস কখনোই নরমাল ডিস্ট্রিবিউশনের জন্য এত সুন্দর একটা ইকুয়েশন তৈরি করতেন না। তার নরমাল ডিস্ট্রিবিউশনের ডেফিনিশন স্ট্যাটিস্টিকাল ক্যালকুলেশন অনেক সহজ করে দেয়। জানিস কি? দুইটা গাউসিয়ানের যোগফল / গুণফলও হয় আরেকটা গাউসিয়ান। তারমানে আমি যদি একটা মান এক নরমাল ডিস্ট্রিবিউশন থেকে নিই, আরেকটা মান আরেকটা নরমাল ডিস্ট্রিবিউশন থেকে নিই; তাদের ক্যালকুলেশনের ফলে আমি আরেকটা ডিস্ট্রিবিউশন পাব কিন্তু সেটাও হবে

নরমাল এবং তার গড় ও ড্যারিয়েন আমি ক্যালকুলেট করেই পেতে পারি। আরও মজার বিষয় হচ্ছে এই মিস্টিড গাউসিয়ান থেকে অনেক কিছু ধারণা করা যায়। তবে প্রথমীতে পিওর গাউসিয়ান বলে কোন কিছু নাই, নরমাল ডিস্ট্রিবিউশন হচ্ছে ন্যাচারাল ডিস্ট্রিবিউশনের একটা অ্যাপ্রক্রিমেশন মাত্র।"

গাউসিয়ান জেনারেশন

এরপর ফেলুদা ঘর থেকে ল্যাপটপটা এনে আমাকে বলল,

-"তোকে আমি এখন হাতে কলমে দেখাব, আমি কি বোঝাতে চাইছি। দেখ,"

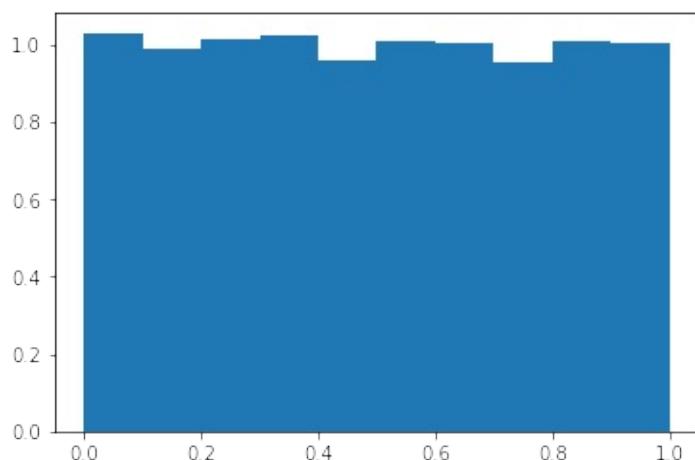
```
N = 1
number_of_students = 10000

u_ = np.random.uniform(0., 1., (number_of_students, N))
mean_ = np.mean(u_, axis=1)

print (u_.shape)

plt.figure()
plt.hist(mean_, bins=10, normed=True)
plt.show()
```

"গাউসিয়ান ডিস্ট্রিবিউশন যেহেতু কষ্টিনিউয়াস রংয়াঙ্গম ড্যারিয়েবলের জন্য হয়, যেখানে হাইট একটা ডিস্ক্রিট ভ্যালু। তাই আমাকে এখানে একটু ক্রিয়েটিভিটি খাটাতে হবে। ধরে নে, তোদের হাইট কষ্টিনিউয়াস এবং এর মান 0 থেকে 1 এর মধ্যে। তাহলে এই 10000 এর মানে হচ্ছে তোদের স্লাসে এতজন স্টুডেন্ট আর $N = 1$ এর মানে আমি একটা স্লাস বিবেচনা করছি। এটা যদি আমি প্লট করি তাহলে গ্রাফ হবে মোটামুটি এরকম।"



-"এবার যদি আমি $N = 2$ বসাই, অর্থাৎ ধর তোদের স্লাসের মত আরেকটা স্লাস কিন্তু সেখানেও 10000 স্টুডেন্ট আছে। তাহলে তার গড় যদি আমি প্লট করি তাহলে গ্রাফ আসবে এমন।"

```

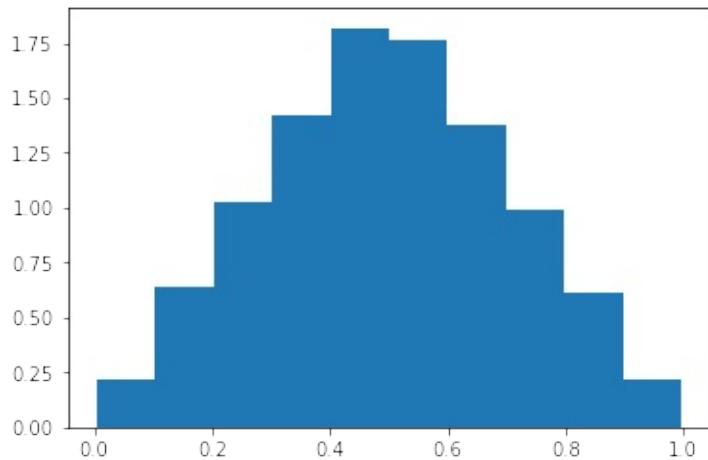
N = 2
number_of_students = 10000

u_ = np.random.uniform(0., 1., (number_of_students, N))
mean_ = np.mean(u_, axis=1)

print (u_.shape)

plt.figure()
plt.hist(mean_, bins=10, normed=True)
plt.show()

```



"ব্যাপারটা বুঝেছিস? আরেকটু পরিষ্কার করলে, ধর তোদের ক্লাসে স্টুডেন্ট আছে তিনজন অর্থাৎ কিনা

number_of_students = 3 এবং যদের হাইট ধরি 5.5, 5.7, 5.8 এবং আরেকটা ক্লাসও আমি বিবেচনায় আনি, যেখানেও স্টুডেন্ট তিনজনই এবং তাদের হাইট 5.6, 5.7, 5.9। তাহলে আমি $N = 2$ এর প্লট টা করব এই ভ্যালু দিয়ে,

$$\frac{5.5+5.6}{2}, \frac{5.7+5.7}{2}, \frac{5.8+5.9}{2}$$

এখন চিন্তা কর number_of_student এবং N এর মান বাড়তে বাড়তে অনেক হ্য তাহলে তার প্লট হবে কীরকম?

আমি চোঁক গিলে বললাম, "গাউসিয়ান ডিস্ট্রিবিউশন?"

-"রাইট! এই দ্যাখ"

```

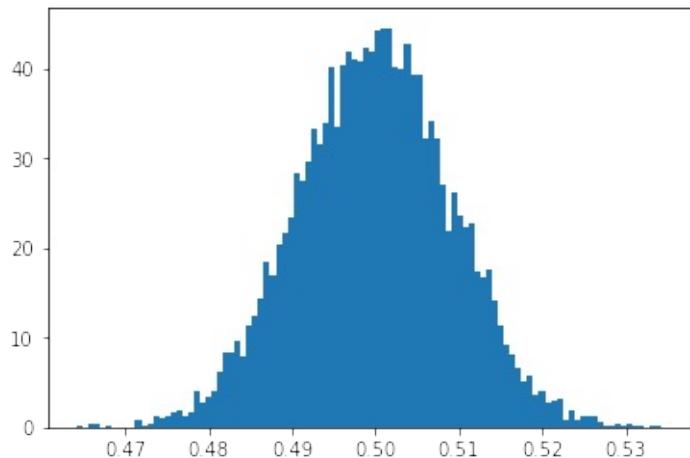
N = 1000
number_of_students = 10000

u_ = np.random.uniform(0., 1., (number_of_students,N))
mean_ = np.mean(u_,axis=1)

print (u_.shape)

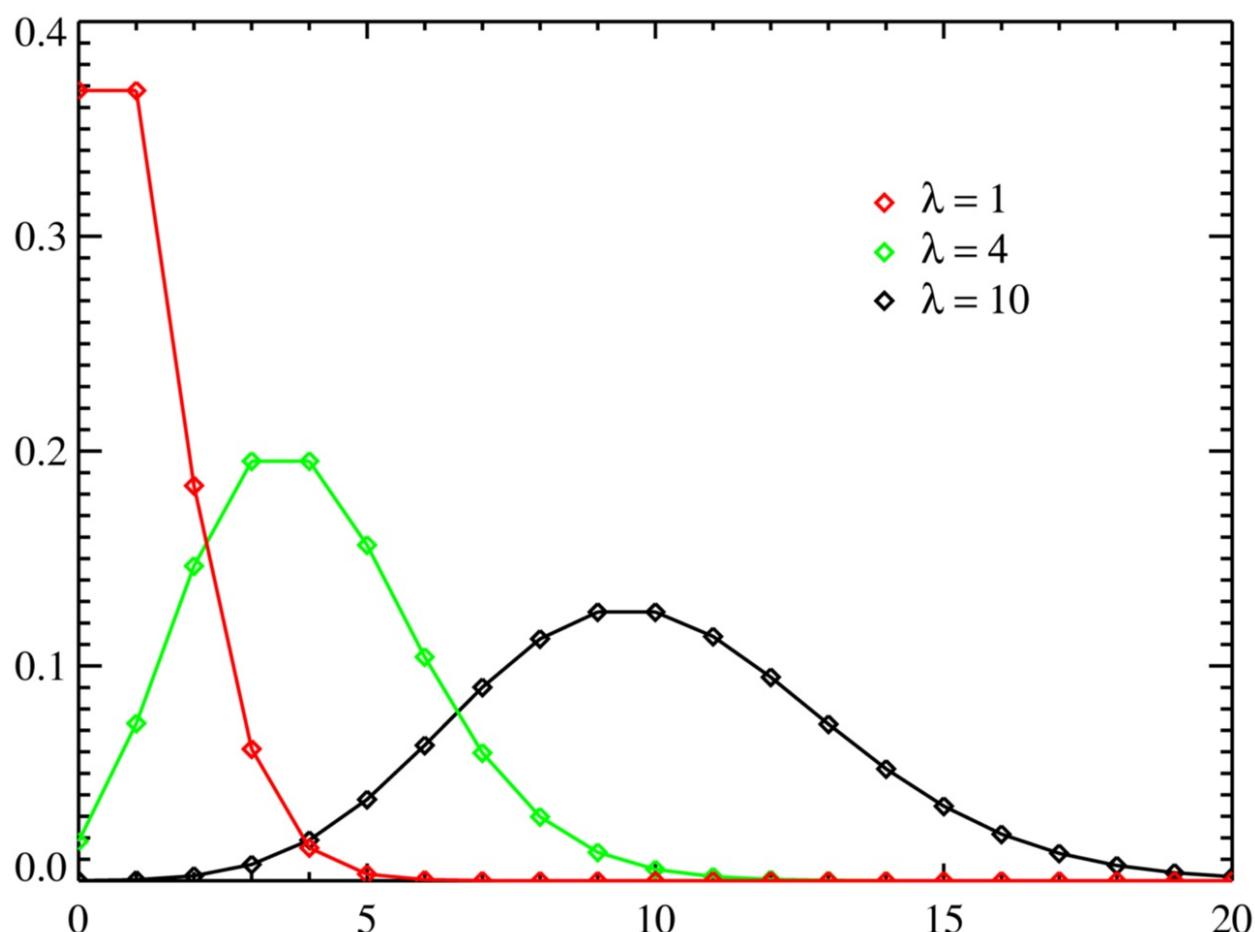
plt.figure()
plt.hist(mean_,bins=100, normed=True)
plt.show()

```



ফেলুদা বলে চলল,

"সবসময় সবকিছুতে নরমাল ডিস্ট্রিবিউশন থাকবে এমন কোন কথা নাই, যেমন চিত্তা কর, কোন একটা রাস্তায় একটা এক্সিডেন্ট হলে সেখানে সতর্কতা বাড়ে এবং কিছু সময় ধরে অ্যাক্সিডেন্টের সংখ্যা কমে আসে। অর্থাৎ অনেকটা এক্সপোনেনশিয়ালি ড্যাম্পড একটা ইকুয়েশন। এটাকে আমরা নরমাল ডিস্ট্রিবিউশনের আওতায় আনলে ক্যালকুলেশন তুল হবে, সেক্ষেত্রে আমরা ব্যবহার করব পয়জন ডিস্ট্রিবিউশন (**Poisson Distribution**)। এভাবে বিভিন্ন ক্ষেত্রের জন্য বিভিন্ন ডিস্ট্রিবিউশন ব্যবহার করতে হয়, আর সেটা ডিপেন্ড করে তুই সমস্যাটা কীভাবে মডেল করবি। বুঝেছিস? λ এর মান বাড়ার সাথে সাথে দেখ পয়জন ডিস্ট্রিবিউশন গাউসিয়ানে রূপান্তরিত হচ্ছে।"



কিছু বুঝেছি কিছু বুঝিনি, বড়্যান্ডম ভ্যারিয়েবল কী জিনিস?! গাউসিয়ানের গড় আর ভ্যারিয়েন্স ই বা কী? কে জানে।

"এই ডিস্ট্রিবিউশন ফাংশনগুলোর পোষাকি নাম প্রব্যাবিলিটি ডিস্ট্রিবিউশন ফাংশন। কন্টিনিউয়াস ডেটার জন্য হয় PDF আর ডিসক্রিট ডেটার জন্য হয় প্রব্যাবিলিটি মাস ফাংশন বা PMF। আরেকটা তথ্য জেনে রাখ,
Distribution Function এর এরিয়া আন্ডার কার্ড সর্বদা ১ হয়। তবে অবজার্ভেশন যদি অনেক বড় হয় বা N এর মান যদি অনেক বড় হয় তাহলে অন্যান্য ডিস্ট্রিবিউশনও নরমাল ডিস্ট্রিবিউশনের মত হয়ে যায়। সেটাতো একটু আগে দেখালামই। তোকে আরেকদিন বিশ্বারিত বলব গাউসিয়ান সম্পর্কে।"

এমন সময় ফোন এল,

ফেলুন্দা ধরে কিছুক্ষণ ওদিকের কথা শুনে বলল,

'সেকী? ডাকাতি?'

...

'আচ্ছা আসছি।'

চল তোপসে, খুনে বাড়িতে এবাব ডাকাতি।

পৌঁছিয়ে অমিতবাবুর সাথে দেখা হতেই বললেন, "মশাই আমি আমাৰ ১০ বছৰেৰ চাকুৱি জীবনে এহেন কাণ দেখিনি। টাকা পয়সা কিছু না, একটা ডায়ারিৰ জন্য একটা মানুষ কীভাবে এতটা মৱিয়া হতে পাৰে?"

"চাকর পঞ্জনন কেমন আছে এখন?"

"ব্রেকফাস্ট করানোর পর একটু সুস্থ এখন। চলুন"

বৈঠকখানায় ঢুকেই পাওয়া গেল পঞ্জননকে। সে ব্যাথায় কাতরাচ্ছে, পুরো কাহিনী এখনো আমার জানা হ্য নি।
ফেলুদার কথাতে মোটামুটি আন্দাজ করতে পারলাম।

"ডাকাত এসেছিল কখন মনে পড়ে?"

"বাবু, আমিতো মেঝেয় ঘুমুচ্ছিলাম, সময় তো দেখিনি। হঠাৎ দেখি জানলার কাছে একটা লাঠি; বইয়ের আলমারির
উপর থেকে কি যেন একটা নামাচ্ছে। নামিয়ে যখন জানলার কাছে নিয়ে আসে আমি ততক্ষণে ঠাহর করতে পারি কি
হ্য। দৌড়ে ধরতে গিয়ে ওর সাথে ধ্বন্তাধ্বনি হ্য। আমি যখন বইটা ধরে টান দিই আমার সাথে কতগুলি কাগজ ছিড়ে
আসে।"

"চেহারা দেখতে পাও নি নিশ্চয়, অবয়বের গড়ন মনে আছে?"

"তা মনে নেই বাবু"

"আচ্ছা, অমিতবাবু; বইয়ের কাগজগুলো পাওয়া যাবে?"

"ওতো আমি বেড়ি করেই বেখেছি। দেখুন।"

আমিও উকি দিয়ে দেখার চেষ্টা করলাম, বুঝলাম না কিছু। হিবিজিবি লেখা, কিছু ডায়াগ্রাম কিছু ইকুয়েশন।

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\therefore \sigma\left(\sum_{i=1}^3 x_i w_i\right) = \frac{1}{1+e^{-(x_1 w_1 + x_2 w_2 + x_3 w_3)}}$$

$$E = (y - \hat{y})^2 \times \frac{1}{2}$$

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial x} \cdot \frac{\partial x}{\partial w}$$

$$\frac{\partial E}{\partial \hat{y}} = 2 \times \frac{1}{2} (y - \hat{y}) = (y - \hat{y})$$

$$\frac{\partial \hat{y}}{\partial x} = \frac{\partial}{\partial x} \left\{ \sigma\left(\sum_{i=1}^3 x_i w_i\right) \right\} = \sigma(x) \{1 - \sigma(x)\}$$

$$\frac{\partial x}{\partial w} = \frac{\partial}{\partial w} \left(\sum_{i=1}^3 x_i w_i \right) = w_1 + w_2 + w_3$$

Iterations?
Local gradient?

What about updates?
weight

Is sigmoid going to work?

শুধু বুঝলাম ডেরিভেশনের চেইন রুল ও পার্শিয়াল ডেরিভেটিভ করা হয়েছে। এর বাইরে কিছু বোধ গেল না।

ফেলুদা অনেক্ষণ ক্র কুঁচকে তাকিয়ে রইল কাগজটার দিকে। Local Gradient, Sigmoid টার্মের সাথে আমি একদম পরিচিত না, ফেলুদাও কি অপরিচিত?

কিছুক্ষণ পরে ফেলুদা বলল, "কাগজটা আমি নিয়ে যাচ্ছি অমিতবাবু!"

অমিতবাবু বললেন, "অবশ্যই। চিঠির ব্যাপারটা নিয়ে কিছু সুবাহা করতে পারলেন?"

ফেলুদা বলল, "সেটা হয়ে গেলে সবার আগে আপনিই জানতে পারবেন। আজ তাহলে চলি, চল তোপসে।" শোনামাত্র ওর সাথে সাথে রওনা দিলাম।

বের হয়ে জিজ্ঞাসা করলাম, "আচ্ছা ফেলুদা সিগময়েড আৱ লোকাল গ্রেডিয়েন্ট কী? কাগজে প্রোবাল গ্রেডিয়েন্ট লেখা নেই কেন?"

-"আমাকে এটা বল, তুই গ্রেডিয়েন্ট বলতে কী বুঝিস?"

-"ফাংশন ডেরিভেটিভ কৰলে যেটা আসে, বাংলায় যাকে বলে ঢাল"

-"গুড়, তাৰমানে লোকাল গ্রেডিয়েন্ট মানে?"

-"তাৰমানে কোন একটা ফাংশনের অনেকগুলো পয়েন্টের গ্রেডিয়েন্টের একটা?"

-"ভেৱি গুড়, আমাৰ তাই মনে হচ্ছে। কিন্তু প্রোবাল গ্রেডিয়েন্ট বলতে কিছু হওয়াৰ কথা না আবাৰ হতেও পাৰে। হ্যাত বইয়েৰ বাকি অংশে এৱ ব্যাখ্যা থাকতে পাৰে যেটা আমাদেৱ হাতে নেই। তুই ট্যাক্সি নিয়ে সটান বাঢ়ি চলে যা। আমি আসছি ঘণ্টাখানেক পৰে।"

কি কৰব ভেবে পাছিলাম না, ভাবলাম গিয়ে হাতে লেখা বাংলা ডিজিটেৱ কাজটা আগাই। ফেলুদা যে বুদ্ধি দিয়েছিল, সেটা কাজে লাগানোৱ দৱকাৰ। মনে পড়ছে কিছু কিছু, ছবি থেকে গুৰুত্বপূৰ্ণ অংশ চিহ্নিত কৰাকে বা যে অংশ গুৰুত্বপূৰ্ণ বলে মনে হয় তাকে *Region of Interest (ROI)* বলে।

যেহেতু ছবিৰ আকাৱ 32×32 তাই ভাবলাম বৰং ছবিকে আমি টোটাল $\frac{32}{3} \times \frac{32}{3}$ এত, কিন্তু আমি তো আৱ ছবিৰ অ্যাৰে ভগ্নাংশেৰ ইলেক্ট্ৰন দিয়ে সিলেক্ট কৰতে পাৰব না। তাই আমাৰ $\frac{32}{3}$ এৱ সবনিষ্ঠ রাউণ্ড মান বা *floor* ব্যবহাৰ কৰতে হবে।

আমি যদি এমন একটা কোড লিখি যেটা একটা ছবিৰ $\frac{32}{3} \times \frac{32}{3}$ বক্স আকাৰে দেখাতে পাৰে তাহলে আমাৰ সেখান থেকে ভ্যালু নিতে সুবিধা হবে। এটা মনে কৰে আমি কোড লেখা শুরু কৰে দিলাম।

```

# Importing necessary libraries
from scipy.ndimage import imread
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from skimage.color import rgb2gray
import matplotlib.patches as patches
import numpy as np
from PIL import Image
import math
from skimage import io, filters, draw

# Loading and converting an image into numpy array
one_array = np.array(Image.open(load_digit(1)[0]), dtype=np.uint8)

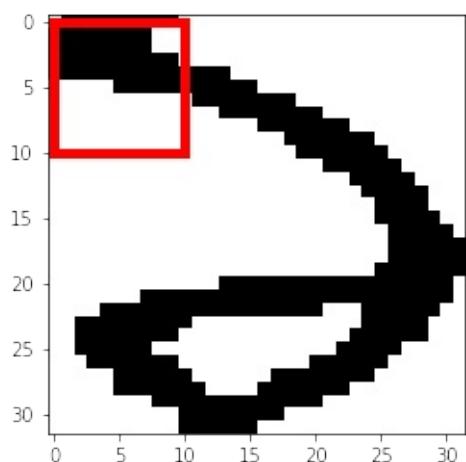
# Getting region of interest [ROI]
height = math.floor(32/3)
width = math.floor(32/3)

for i in range(3):
    for j in range(3):
        fig, ax = plt.subplots(1)
        ax.imshow(one_array)
        print("Patch Coordinates: {}, {}".format(j, i))
        rect = patches.Rectangle((height * i, width * j), height, width, linewidth=5,
edgecolor='r', facecolor='none')
        ax.add_patch(rect)
        plt.show()

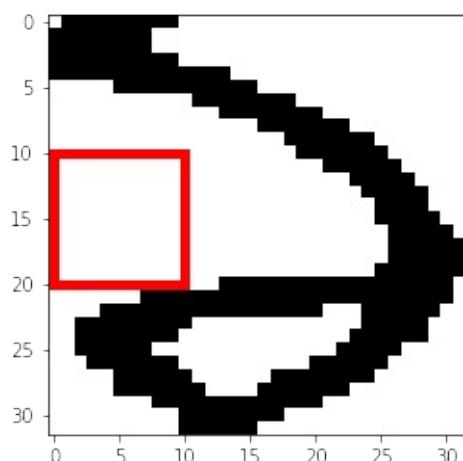
```

ଆଟ୍ରିପ୍ଟ ଆସଲ ଏଟା,

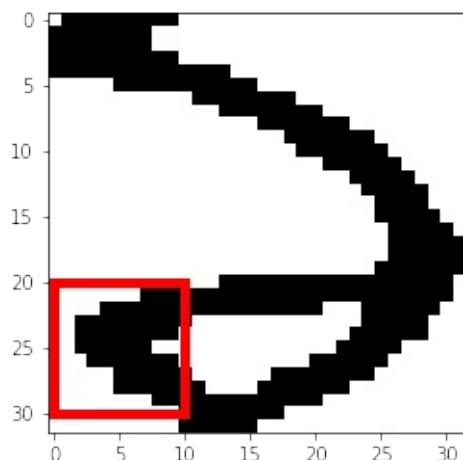
Patch Coordinates: 0, 0



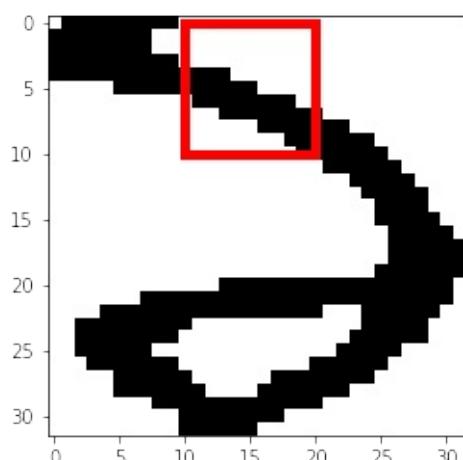
Patch Coordinates: 1, 0



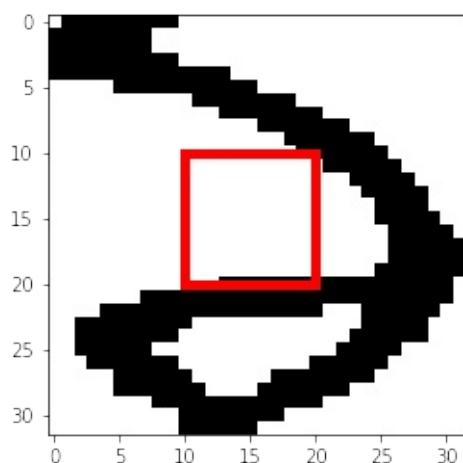
Patch Coordinates: 2, 0



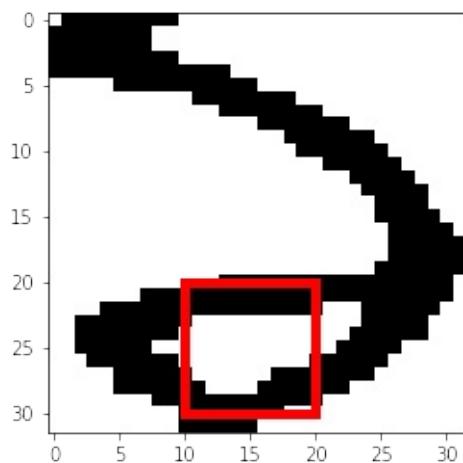
Patch Coordinates: 0, 1



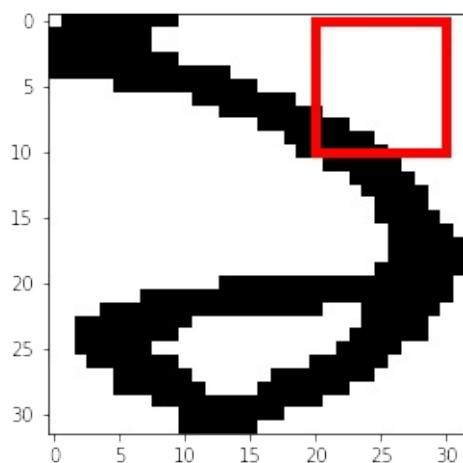
Patch Coordinates: 1, 1



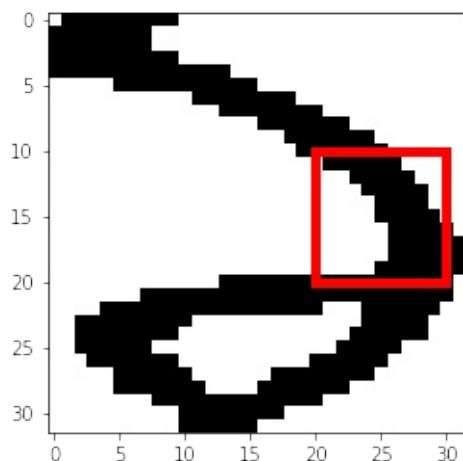
Patch Coordinates: 2, 1



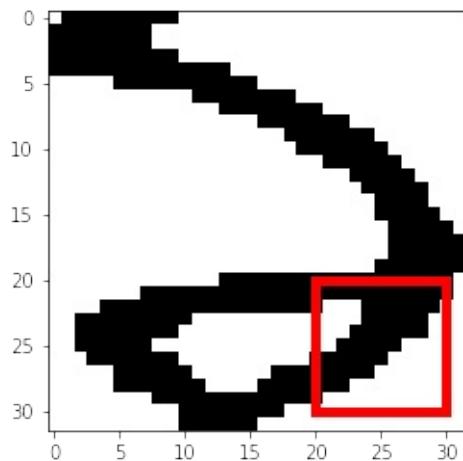
Patch Coordinates: 0, 2



Patch Coordinates: 1, 2



Patch Coordinates: 2, 2



যেহেতু এটা বারংবার ব্যবহার করব তাই এটাকে ফাঁশন বানানোই উচিং বলে মনে হল।

```
def draw_roi(digit, index):
    image_array = np.array(Image.open(load_digit(digit)[index]), dtype=np.uint8)
    height = math.floor(32/3)
    width = math.floor(32/3)

    for i in range(3):
        for j in range(3):
            fig, ax = plt.subplots(1)
            ax.imshow(image_array)
            print("Patch Coordinates: {}, {}".format(j, i))
            rect = patches.Rectangle((height * i, width * j), height, width, linewidth=5, edgecolor='r', facecolor='none')
            ax.add_patch(rect)
            plt.show()
```

এবার আমি সহজেই যে ডিজিট ও যত নম্বর ছবির রিজিওন দেখতে চাই এই ফাঁশন করলেই হবে।

এখন সবচেয়ে গুরুস্থপূর্ণ কাজ, ROI এ পিঙ্কেল ডেনসিটি মেজার করতে হবে এবং পজিশনওয়াইজ ডেনসিটি কত হয় তার একটা হিসেব রাখতে হবে। যেমন শুন্য এর ক্ষেত্রে $(1, 1)$ পজিশনে পিঙ্কেল ডেনসিটি কম হওয়ার কথা কিন্তু চারিদিকে কিছু না কিছু থাকার কথা। আবার ১ এর ক্ষেত্রে $(0, 1), (1, 1)$ পজিশনে ডেনসিটি কম হবে কিন্তু অন্যান্য ক্ষেত্রে বেশি থাকবে। এভাবে আমি কিছু রুল তৈরি করে সহজেই ক্লাসিফাই করতে পারব।

দরজায় কড়া নেড়ে উঠল, আমি গিয়ে দরজা খুলতেই দেখি ফেলুদা। সটান ঢুকে সোফায় গা এলিয়ে দিয়ে বলল। চা বলে দিয়ে আয়তো। চায়ের কথা বলে ফেলুদার সামনে বসলাম। এরমধ্যে ফেলুদা একটা চারমিনার ধরিয়ে ধোঁয়ার রিং ছাঢ়তে ছাঢ়তে বলল, "বল কী জানতে চাস!"

বাহ, মেঘ না চাইতেই জল, সচরাচর এমনটা হয় না। সুযোগটা আমি লুফে নিয়ে বললাম, "ডায়ারিটা কেন খুনের দিন নিল না?"

-"তুই বুঝলি কী করে খুন যে করেছে ডাইরিও সে নিয়েছে?"

এই বে, এটা তো মাথায় আসে নি। ফেলুদা বলল,

-"এখানেই বিষয়টা অত্যাধিক জটিল হয়ে যাচ্ছে। যে খুন করেছে কিন্তু খুনের দিন ডায়ারি নেয় নি। ধরি খুনি X , তাহলে যে ডায়ারি নিয়েছে সে হল Y , প্রশ্ন হচ্ছে, Y ও X পরস্পরকে চেনে কিনা? আরও বড় প্রশ্ন হচ্ছে, X চিঠি পাঠিয়েছে নাকি Y ? নাকি X এর খুনের সুযোগ নিয়ে Y স্বার্থোধ্বার করল?"

-"তুমি এতক্ষণ কোথায় ছিলে?"

ফেলুদা সেকথার জবাব না দিয়ে বলল, "তোর কাজ কতদূর?"

-"এগোচ্ছে, আমি পজিশন ভেদে পিঙ্কেল ডেনসিটি মাপতে যাচ্ছিলাম। আচ্ছা ফেলুদা, কাগজে যে হিবিজিবি লেখা ছিল, তার কিছু বুঝতে পেরেছ?"

-"সেটা বোধার জন্যই বেরিয়েছিলাম, দুইজন ম্যাথেমেটিশিয়ানের সাথে দেখা করে, তিনটে লাইব্রেরি ঘুরে আর একজন নিউরোসায়েন্টিস্টের সাথে দেখা করে এলাম।"

-"নিউরোসায়েন্টিস্ট?! এখানে নিউরোসায়েন্স আসলো কোথেকে? ম্যাথেমেটিশিয়ানের ব্যাপার না হ্য বুঝলাম।"

-"তোর অত বুদ্ধি থাকলে আমার কাজটা তুইই করতি, এবার শোন নিউরোসায়েন্টিস্ট থেকে অঙ্গুত সব তথ্য পেলাম তবে সেটার ধারণা পাই লাইব্রেরি থেকে, বললে চিন্তাও করতে পারবি না জিনিসটা এত প্যাঁচালো, যা জানলাম সেটা হল...."