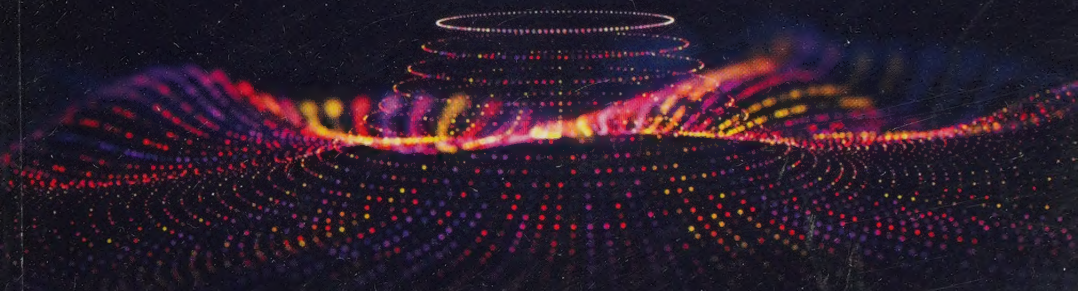
HEARD IN DATA SCIENCE INTERVIEWS

Over 650 Most Commonly Asked Interview Questions & Answers



KAL MISHRA

KAL MISHRA

HEARD IN DATA SCIENCE INTERVIEWS

OVER 650 MOST COMMONLY ASKED INTERVIEW QUESTIONS & ANSWERS

This paperback edition first published in 2018

ISBN 978-1-7272-8732-5

To my parents

Contents

List of Figures

List of Tables

Introduction

This book is the outgrowth of my notes of AI technical interview questions accumulated over several years. It contains over 650 questions from actual job interviews for Data Scientist/Machine Learning Engineer positions and provides full answers for them. Using this list an interviewee should be able to prepare for the most expected types of questions, learn what answers will positively resonate with employers, and develop confidence to succeed in the interview.

The questions were collected by me from interviewees, interviewers, and others based in Silicon Valley and other metro areas in the United States, but should be equally relevant for AI expert positions elsewhere. In addition, I have included questions I tend to ask as an interviewer and also those that I faced as an interviewee. Some of these questions may come up during phone screening, but most are likely to be asked in on-site interview sessions.

I have tried my best to only keep genuine AI questions and left out those that seemed "fizz-buzz" type, "prove a theorem" pedantic, "explain *<insert your favorite topic here>* to me like I'm a 5 year old" teasers, or plain old gotcha questions. I believe such questions are insulting to the candidate's intelligence, since knowing their answers has no correlation with being a good data scientist.

How to Use This Book

If you are an applicant preparing for data science interviews, go through the questions and attempt to answer them first without looking to the clues at the back of the book. Answers in the appendix

are typically very short and to-the-point. You should study topics covered by these questions in full detail by consulting the references I give. Don't let the long list of questions discourage you, especially if you do not know the answers to some of them. Very few jobs actually require applicants to know all of the topics in detail, so it is unlikely that a significant number of these questions will be asked during one particular interview. The best use of the questions in this book is to re-familiarize yourself with the ideas and techniques you have already learned before.

If you are an interviewer searching for a standard set of questions to ask, you can certainly use these questions as they stand. However, you can get more creative by modifying the question in one of the following ways: (a) Change the premise of the question, e.g., from *when should you use Naïve Bayes* to *when should you NOT use Naïve Bayes*, or from *maximize X* to *minimize X* etc., (b) Use different numbers in questions like 15^18^15 =?, (c) Ask follow up questions or probe with specific examples for the sake of clarity, and (d) If the question involves assumptions, you can relax or tighten these to probe the candidate's depth of understanding.

Choice of Topics

Data science is a fast-evolving field, so interview for AI jobs may include a broad range of questions. The choice of topics in this book is aimed at covering the most frequently asked questions. Below I give brief overview of the topics covered in each chapter and how to prepare for them.

I **Computer Science Fundamentals**
Most questions in this vein are traditional Computer Science topics, such as big $\mathcal{O}$ notation, linear and binary search, sorts, data structures, programming practices, etc. The interviewers will test your skill in several different ways. They may ask you to solve a problem in theory without writing actual code. They may also give you whiteboard exercises to code on the spot. Sometimes the latter is replaced by a take-home coding assignment.

To answer programming questions, it is important that you first discuss with the interviewer to outline a simple-but-working solution. Try to get the basic data structure and algorithm right, even if your solution is sub-optimal. Clearly state any assumptions and tradeoffs made. At this step, you may want to write quick pseudo-code, specifically outlining the edge cases and time/space complexity. Some interviewers want to talk about system design, architecture, and scaling up challenges, which is your opportunity to demonstrate knowledge of these topics.

Finally, depending on the interviewer's interest and available time, write any real code in the language you are most comfortable with. Writing code on whiteboard is inherently difficult because you are outside of your comfort zone and can't look anything up. No one codes on a whiteboard outside of interviews! So, make sure to practice beforehand.

II **Probability & Statistics**

Probability and statistics are the twin foundations of Machine Learning. It is difficult to succeed as a data scientist without an advanced knowledge of both. Accordingly, it is likely that the interviewers will probe your understanding of the subject matter with questions related to conditional probability, Bayes' theorem, maximum likelihood, experiment design, sampling, hypothesis testing (including A/B testing), common probability distributions, etc.

The key to successfully answering these questions is to communicate with clarity. As much as possible, try very simple mathematics to explain basic ideas. Use case studies and practical examples if you can.

III **Machine Learning**

Machine Learning (ML) lies at the heart of data science, both in exploratory analysis and in building predictive models. The interviewers typically ask three types of ML questions. The first is about your general interest in ML, e.g, what's going on in the industry, what's the latest trend, how would you apply ML to some popular task *X*, etc. The second type is mostly about algorithms and theory — supervised vs. unsupervised learning, classification, regression, anomaly detection, Deep Learning, rec-

ommender systems, etc. The third type is about the application of those algorithms — bias-variance tradeoff, performance metric, precision vs. recall, missing data, etc.

You are expected to have strong familiarity with at least one ML algorithm. Make sure you know the implementation details, use cases, advantages, and limitations of that particular algorithm. Also, be ready to describe a production-ready ML solution you implemented on a real dataset in the past.

IV **Data Modeling**

Data modeling is where a data scientist provides most value for the company. However, a significant amount of effort also goes into the collection, wrangling, and cleaning of the data. Ultimately, we draw conclusions from data that help us solve a particular problem. Interviewers often ask how you would model data in specific hypothetical situations. They may ask you to dive deeper to explain how you would implement your model given a set of constraints. As a follow up, it is common to ask statistics-oriented questions regarding the validity of the model itself or underlying assumptions.

You already know that turning data into predictive and actionable information is difficult. Talking about it to a potential employer is even more so. Therefore, it is important that you do a lot of practice before the interview describing your past model building experience. In particular, details about the techniques used, challenges overcome, and successes achieved during the process. Do not hesitate to draw diagrams and flowcharts. Always emphasize practice over theory!

V **Natural Language Processing**

With the advent of digitization, most of the data being created these days happen to be unstructured text, video, and audio. Natural Language Processing (NLP) plays an important role in making sense of these unstructured data. Therefore, depending on the interviewing company's domain, you are likely to get some NLP questions during the interview. Typical topics include bag of words, sentence segmentation, tokenization, stemming, word embeddings, named entity recognition, sentiment analysis, topic modeling, etc.

If you used NLP in a significant way in one of your past projects, be ready to describe it in simple terms. If you didn't, make yourself familiar with the basic NLP concepts and at least try a simple document classification exercise before going to the interview.

VI **Distributed Systems and Big Data**
This is a diverse field, but as a general rule, you should have good understanding of the major components in a typical data pipeline and be able to assess the pros and cons of different tools for specific tasks. Refresh your knowledge of relational databases, CAP theorem, SQL, and at least one modern NoSQL database. You are expected to have some hands-on experience with large, real-world data in the cloud using Hadoop–Spark framework.

In addition to general questions on the above topics, the interviewers may also ask you about core concepts of distributed file system and Spark — such as resilient distributed data stores, memory caching, actions, transformations — and distributed machine learning. Spend some time practicing simple SQL and Spark queries. Get acquainted with popular cloud computing tools so that you are comfortable talking about them and potentially using them in next projects.

VII **Culture-fit Questions**
At some stage during the interview process you are likely to face culture-fit questions. Through these questions the interviewer is trying to get a sense of how you would fit within the company and also gauge your interest in joining their team. Typical topics include teamwork, leadership, conflict management, problem solving, failures, and other such anecdotes drawn from past experience.

There are no right or wrong answers to these, but the best answers are communicated with confidence and a smile. Before the interview, write down examples of work experience related to these topics to refresh your memory. You will need to recall specific examples.

Do not forget that as a candidate you are also reviewing the company. You will be dedicating several years of your life to

this workplace. So, make sure to use the interview process to figure out if this is a right kind of place for you. In particular, their team culture and how much the company values Machine Learning and AI.

Acknowledgments

I wish to thank all my friends and colleagues who supported me during the writing of this book through useful comments and suggestions. Special thanks to those who supplied me with information and questions. Any errors and omissions are obviously mine alone.

Kal Mishra
Menlo Park, California
August 31st, 2018

Chapter 1
Computer Science Fundamentals

Question 1.1 Write a function to convert a string into integer.

Question 1.2 Write a function to compute Fibonacci numbers recursively. What is its time complexity?

Question 1.3 Write a program to roll a dice 1000 times and display how many times each number was rolled.

Question 1.4 How would you reverse the order of words in a string in place? For example, if the input string is "i like this program very much", the function should return "much very program this like i".

Question 1.5 Given a positive number N, efficiently generate all binary numbers between 1 and N.

Question 1.6 You are given the arrival and departure times of airplanes at an airport for a single day. Schedules for the airplanes remain the same across all days. For example, *Arrival* = [9:30, 11:15, 16:30], *Departure* = [11:45, 11:30, 16:45]. Each array is sorted by time. The *Departure* array is sorted by corresponding arrival times (i.e., plane i arriving at time $Arrival[i]$ departs at $Departure[i]$). Determine the minimum number of gates the airport should have so that no plane spends time waiting for a gate. A gate can be considered occupied at the arriving minute and empty at the departing minute.

Question 1.7 A message containing letters from A to Z is being encoded to numbers using the following mapping: A $\rightarrow$ 1, B $\rightarrow$ 2, ..., Z $\rightarrow$ 26. Given an encoded message containing digits, how will you determine the total number of ways to decode it? For example, if input = 1234, the possible decodings are: ABC, LC, AW. You can assume that the input contains only valid digits from 0 to 9. There are no leading 0's, no extra trailing 0's, and no two or more consecutive 0's.

Question 1.8 You are given a word document. How will you print all the unique words in the document with frequencies? How will you find the top k ranked words along with their frequency?

Question 1.9 What is the difference between big-endian and little-endian?

Question 1.10 What is left shifting and right shifting a number?

Question 1.11 How would you write -5 in binary?

Question 1.12 How can you quickly compute 2^x?

Question 1.13 How can you quickly determine whether a number is a power of 2?

Question 1.14 What is the value of 15^18^15?

Question 1.15 Write a code in your favorite programming language to count number of 1's in a binary representation.

Question 1.16 Write a code in your favorite programming language to count number of trailing zeros in a number.

Question 1.17 How many unique subsets of n different objects can we make?

Question 1.18 What is your favorite programming language? Why?

Question 1.19 Describe a data science project in which you worked with a substantial programming component. What did you learn from that experience?

Question 1.20 All numbers in an array with length $n+1$ are in range from 1 to n, so there is at least one duplication in the array. How would you find the duplication? Please don't modify the input array.

Question 1.21 What is polymorphism?

Question 1.22 What is a virtual function (in C++)?

Question 1.23 What is the difference between Stack and Queue data structures?

Question 1.24 What is the time complexity to add, remove, and look up an element in a linked list?

Question 1.25 How would you check if a linked list has cycles?

Question 1.26 Given two elements in a binary tree, how would you find their lowest common ancestor?

Question 1.27 Design an algorithm to find the path from one node in a binary tree to another.

Question 1.28 What is a binary search tree? What is the time complexity for insertion, removal, and lookup for an element?

Question 1.29 Find the second largest element in a binary search tree.

Question 1.30 What is the time complexity of any comparison-based sorting algorithm? Can you prove it?

Question 1.31 Describe what is a merge sort.

Question 1.32 Describe quicksort.

Question 1.33 How would you sort a large list of numbers?

Question 1.34 Let's say you have an array containing information regarding n people. Each person is described using a string (their name) and a number (their position along a numbered line). Each person has three friends, whose numbers are adjacent to the person's number. Describe an algorithm to identify each person's three friends.

Question 1.35 Describe an algorithm to identify the k^{th} smallest element in an array of n elements.

Question 1.36 How will you find the median of an array of n elements?

Question 1.37 How would you sort a stack such that the largest element is at the top?

Question 1.38 Given a sorted array of integers, how can you find the location of a particular integer x?

Question 1.39 How will you find the shortest path from one node to another in a weighted graph? What if some weights are negative?

Question 1.40 How will you find all palindromic substrings in a given string?

Question 1.41 Have you optimized code or algorithms: in C++, Python, Java, Scala, etc. How, and by how much?

Question 1.42 What are *VTABLE* and *VPTR* in C++?

Question 1.43 Describe a difficult bug you've encountered and how you resolved it.

Question 1.44 How will you compute the inverse of a matrix faster by playing around with some computational tricks?

Question 1.45 What is hashing? Give an example of when you might want to use it. When should you not use it?

Question 1.46 What are hash table collisions? How can they be avoided? How frequently they occur?

Question 1.47 Is it better to have 100 small hash tables or one big hash table, in memory, in terms of access speed (assuming both fit within RAM)?

Question 1.48 Given an English word in the form of a string, how can you quickly find all valid anagrams for that string (all valid re-arrangements of the letters that form valid English words)? You are allowed to pre-compute whatever you want to and store whatever you optionally pre-compute on disk.

Question 1.49 Given two valid dictionary words of same length, write a function which returns the minimum number of steps to go from the first to the second word. There are two conditions: you can change only one character at a time and the word formed at each step should be a valid dictionary word. For example, the minimum number of steps to go from "cat" to "dog" is: cat $\rightarrow$ cot $\rightarrow$ dot $\rightarrow$ dog, so 3 steps. You are allowed to pre-compute whatever you want to and store whatever you optionally pre-compute on disk.

Question 1.50 What is a multi-threaded program?

Question 1.51 What is mutex?

Question 1.52 What is semaphore?

Question 1.53 What is a deadlock in the context of a multi-threaded program? How can we ensure that deadlock does not occur?

Question 1.54 How would you rotate a matrix by 90 degrees?

Question 1.55 How would you rotate a matrix by 180 degrees?

Question 1.56 Suppose you are analyzing streaming data and want to store just the current values of the mean and standard deviation of an observable. You just got a new data point. How would you update your information?

Question 1.57 What is a RESTful API?

Question 1.58 What is REPL?

Question 1.59 Describe stateful and stateless algorithms. What are their advantages and limitations?

Question 1.60 How would you check if an IPv4 address is legal?

Question 1.61 What is Occam's Razor?

Question 1.62 What is Moore's law?

Question 1.63 What do you think about in-database analytics?

Question 1.64 What is a cron job?

Question 1.65 What is dynamic programming? How's it different from recursion?

Question 1.66 What are software patterns? Which patterns are you familiar with?

Question 1.67 When you might want to use a singleton pattern? Write example of a singleton class implementation.

Question 1.68 What do you know about Model View Controller?

Question 1.69 Write a code in your favorite programming language to check if parentheses in a string are balanced.

Question 1.70 Write a code in your favorite programming language to implement a person class with attributes like name, address, gender, age, etc.

Question 1.71 How does floating point affect precision of calculations?

Question 1.72 You are given a text file. Write a command to find the count of lines containing the word "ABC".

Question 1.73 What is the difference between *grep* and *egrep*?

Question 1.74 Given a file, replace all occurrence of word "ABC" with "DEF" from the 5^{th} line till end in only those lines that contains the word "MNO".

Question 1.75 Write a command to count the total number of fields in a file.

Question 1.76 Write a command to print the even-numbered lines in a file.

Question 1.77 Given a file, write a command sequence to find the count of each word.

Question 1.78 How will you find the 99^{th} line of a file using only *tail* and *head* commands?

Question 1.79 Write a command to print the 10^{th} line from a file without using *tail* and *head* commands.

Question 1.80 How can we delete all blank lines in a file?

Question 1.81 How can I insert a line "ABCDEF" at every 100^{th} line of a file?

Question 1.82 I have two files and I want to print the records which are common to both. How can I do this via command line?

Question 1.83 How can you find out how long the system has been running?

Question 1.84 I want to monitor a continuously updating log file. What command can be used to most efficiently achieve this?

Question 1.85 How will you find the total disk space used by a specific user?

Question 1.86 How can we find the process name from its process id?

Question 1.87 Are you familiar with version control? What tools and processes have you used for this?

Question 1.88 Have you ever worked in a developer team that followed a particular agile process?

Question 1.89 What is a technical debt? How can one mitigate it? How relevant is this to deploying data driven models in the real world?

Question 1.90 How do you test your code? What kind of tests do you write?

Question 1.91 What is "Fermat's Last Theorem"?

Question 1.92 You are given a pseudo random number generator. How would you test that the generated numbers are truly random?

Question 1.93 What is the difference between deterministic and probabilistic data structures? Are you familiar with any probabilistic data structure?

Question 1.94 What is Bloom filter? How can it be used to query whether an element exists in a large set?

Question 1.95 What is HyperLogLog? How is it used to estimate the cardinality of a very large data set?

Question 1.96 What is locality-sensitive hashing? Give me one use case where you would use it.

Question 1.97 What is POC (proof of concept)?

Question 1.98 What is the size of an integer on a 32-bit system?

Question 1.99 Write a regular expression that can match any web URL.

Question 1.100 How will you detect individual paid accounts shared by multiple users?

Question 1.101 Name a few famous APIs.

Question 1.102 How do you optimize algorithms? Provide some examples.

Chapter 2

Probability and Statistics

Question 2.1 Mr. Smith has two children. You already know that one of them is a girl. What's the probability that Mr. Smith has one boy and one girl?

Question 2.2 In the classic Pick-6 lottery game, six numbers are drawn from a set of 49. What is the probability to win a Pick-6 lottery?

Question 2.3 Given two fair dices, what is the probability of getting scores that sum to 4? To 8?

Question 2.4 We have got 4 medicine bottles each containing 10 pills. Each pill is supposed to have 10 mg weight, but one of the bottles is defective and contains pills of lighter weight. How will you find the defective bottle by weighing only once?

Question 2.5 What is the expected number of fair coin flips to get two consecutive heads?

Question 2.6 What is the expected number of fair coin flips to get three consecutive heads? What about n consecutive heads?

Question 2.7 A fair coin flip experiment is carried out N times. What is the expected number of heads?

Question 2.8 If you repeatedly flip a coin whose probability of heads is p, then what is the expected number of flips you need to do in order to get a head immediately followed by a tail?

Question 2.9 Without using a calculator, how many zeros are at the end of 100!?

Question 2.10 What is the minimum number of people we would need to assemble in a group such that the probability that at least one person in the group has the same birthday as you is greater than 50%?

Question 2.11 You are trapped in a dark cave with three indistinguishable exits on the walls. One of the exits takes you 3 hours to travel and takes you outside. One of the other exits takes 1 hour to travel and the other takes 2 hours, but both drop you back in the original cave through the ceiling, which is unreachable from the floor of the cave. You have no way of marking which exits you have attempted. What is the expected time it takes for you to get outside?

Question 2.12 Imagine a test with a true positive rate of 100% and false positive rate of 5%. Imagine a population with a $1/1000$ rate of having the condition the test identifies. Given a positive test, what is the probability of having that condition?

Question 2.13 The mean heights of men and women in a population were calculated to be μ_M and μ_W. What is the mean height of the total population?

Question 2.14 A recent poll revealed that a third of the cars in Italy are Ferraris, and that half of those cars are red. If you spot a red car approaching from a distance, what is the likelihood that it is a Ferrari?

Question 2.15 You're about to get on a plane to Seattle. You want to know if you should bring an umbrella. You call 3 random friends of yours who live there and ask each independently if it's raining. Each of your friends has a $2/3$ chance of telling you the truth and a $1/3$

chance of lying. All 3 friends tell you that "Yes" it is raining. What is the probability that it's actually raining in Seattle? Assume prior probability that it's raining on any given day in Seattle is 25%.

Question 2.16 You are given two eggs, and access to a 100-story building. Both eggs are identical. The aim is to find out the highest floor from which an egg will not break when dropped. If an egg is dropped and does not break, it can be reused. If the egg breaks when dropped from floor n, then it would also have broken from any floor above that. If the egg survives a fall, then it will survive any fall shorter than that. The question is: What strategy should you adopt to minimize the number of egg drops it takes to find the solution? And what is the worst case for the number of drops it will take?

Question 2.17 What is a Bernoulli trial? What is the probability of k successes in n Bernoulli trials?

Question 2.18 What is the Binomial Probability Formula?

Question 2.19 Suppose that you toss a balanced coin 100 times. What is the approximate probability that you observe less than or equal to 40 heads?

Question 2.20 Find the probability of getting 3 left-handed students in a class of 15 students, given that 10% of students are left-handed.

Question 2.21 What is the Poisson Probability Formula?

Question 2.22 Let's assume a big earthquake strikes San Francisco bay area once every 100 years. Calculate the probability of $k = 0, 1, 2, 3$ earthquakes occurring in a 100-year interval.

Question 2.23 A bakery makes a batch of 200 cookies in which 2000 chocolate chips were used. What is the probability that a cookie picked at random from the batch will contain at least 13 chocolate chips?

Question 2.24 Of the nine members of the board of trustees of a college, five agree with the president on a certain issue. The president selects three trustees at random and asks their opinions. What is the probability that at least two of them will agree with him?

Question 2.25 You have an array in which the i^{th} element is the price of a given stock on day i. If you were only permitted to buy one share of the stock and sell one share of the stock, design an algorithm to find the best times to buy and sell.

Question 2.26 You have 8 coins which are all the same weight, except for one which is slightly heavier than the others. You don't know which coin is heavier. You also have an old-style balance, which allows you to weigh two piles of coins to see which one is heavier. What is the fewest number of weightings needed to tell which coin is the heavier one?

Question 2.27 You have 1000 coins, one of which is faulty: It has a head on both sides. You randomly draw a coin, and without examining it, toss it 10 times. As it happens, you get 10 heads in a row. What's the probability that it's the faulty one?

Question 2.28 An ant and a blind spider are on opposite corners of a cube. The ant is stationary and the spider moves at random from one corner to another along the edges only. What is the expected number of turns before the spider reaches the ant?

Question 2.29 What is the Law of Large Numbers? Why is it important in statistics?

Question 2.30 What is the Central Limit Theorem and why is it important?

Question 2.31 What are some pros and cons of your favorite statistical software?

Question 2.32 What summary statistics do you know?

Question 2.33 What is the standard deviation of $(1, 2, 3, 4, 5)$?

Question 2.34 In a game the player tosses a fair die once and the payoff is \$1 for each dot on the upturned face. What is his expected gain?

Question 2.35 What is a normal distribution? Give example of a variable that follows this distribution.

Question 2.36 What is a log-normal distribution?

Question 2.37 How will you check if a distribution is close to normal?

Question 2.38 Given two normal distributions with μ_1, σ_1 and μ_2, σ_2, how would you check if they overlap or not?

Question 2.39 Given two normal distributions with μ_1, σ_1 and μ_2, σ_2, how would you compute the percentage of overlap between them?

Question 2.40 Is the sum of two independent Gaussian distributions also a Gaussian? How? How about the product of two independent Gaussians?

Question 2.41 What is an example of a dataset with a non-Gaussian distribution?

Question 2.42 What is a (Student's) t-distribution? How is it different from the normal distribution?

Question 2.43 What is an F-distribution?

Question 2.44 Explain what a long tailed distribution is and provide three examples of relevant phenomena that have long tails. Why are they important in classification and prediction problems?

Question 2.45 Do you know what the exponential family is?

Question 2.46 A pain-relief drug is tested on 1500 human subjects, and no adverse event is recorded. What is the 95% confidence upper bound on the percentage of people that will experience an adverse event?

Question 2.47 Give examples of data that have very chaotic distribution.

Question 2.48 Do you know the Dirichlet distribution? The multinomial distribution?

Question 2.49 Do you know a few rules of thumb used in statistical or computer science?

Question 2.50 What is the difference between covariance and correlation?

Question 2.51 Is it possible to capture the correlation between continuous and categorical variables? If yes, how?

Question 2.52 What are confounding variables?

Question 2.53 What is the curse of big data?

Question 2.54 What is statistical power?

Question 2.55 What is root cause analysis? How will you identify a cause vs. a correlation?

Question 2.56 What do you understand by Type I vs. Type II error? Alternatively, explain what precision and recall are. How do they relate to the ROC curve?

Question 2.57 Please describe confusion matrix in simple terms.

Question 2.58 Is it better to have too many false positives, or too many false negatives? Explain.

Question 2.59 Given n samples from a uniform distribution $[0, d]$, how would you estimate d?

Question 2.60 You are given an array of distinct integers. Devise an algorithm to randomly reorder the integers such that each possible reordering is equally likely. In other words, given a deck of cards, how can you shuffle them such that any permutation of cards is equally likely?

Question 2.61 Why do we need hypothesis testing?

Question 2.62 What is the null hypothesis? How do we state it?

Question 2.63 What is a p-value? Would your interpretation of p-value change if you had a different (say, much bigger) dataset?

Question 2.64 What is the distribution of p-values, in general?

Question 2.65 What is t-Test/F-Test/ANOVA? When to use it?

Question 2.66 How would you test if two populations have the same mean? What if you have 3 or 4 populations?

Question 2.67 You applied ANOVA and it says that the means are different. How will you identify the populations where the differences are significant?

Question 2.68 What is a confidence interval and why is it useful?

Question 2.69 What is a point estimate? What is a confidence interval for it? How are they constructed?

Question 2.70 How would you build non-parametric confidence intervals, e.g., for scores?

Question 2.71 Why is randomization important in experimental design?

Question 2.72 How will you calculate needed sample size? What is power analysis?

Question 2.73 Are you familiar either with extreme value theory, Monte Carlo simulations, or mathematical statistics (or anything else) to correctly estimate the chance of a very rare event?

Question 2.74 What is the difference between extrapolation and interpolation?

Question 2.75 Why do we need to sample and how?

Question 2.76 When you sample, what bias are you inflicting?

Question 2.77 What is selection bias? Why is it important? How do you control for biases? What are some of the first things that come to mind when you do "*X*" in terms of biasing your data?

Question 2.78 Why do we ever need to sample from a distribution? What are some popular sampling techniques?

Question 2.79 What is accept-reject sampling?

Question 2.80 What is Gibbs sampling?

Question 2.81 What is Metropolis-Hastings sampling?

Question 2.82 What is importance sampling?

Question 2.83 What is the difference between Metropolis-Hastings and Gibbs sampling methods?

Question 2.84 Why importance sampling is so widely used in Monte Carlo simulations? What ensures that the importance sampling estimator is unbiased?

Question 2.85 If you need to sample from a high dimensional distribution, would you use accept-reject method or a Markov Chain Monte Carlo method — such as Metropolis-Hastings? Why?

Question 2.86 Why is it that in Markov Chain Monte Carlo simulation a large number of initial samples need to be thrown away ("burn-in period")?

Question 2.87 Explain what resampling methods are and how they differ. Also explain their limitations. What is bootstrapping?

Question 2.88 How will you use resampling for hypothesis testing? Have you heard of Permutation Tests?

Question 2.89 We have a data sample of body temperatures of healthy adults: $n = 106$, $\bar{x} = 98.20°\text{F}$, $\sigma = 0.62$. For a 0.05 significance level, test the claim that the mean body temperature of healthy adults is equal to 98.6°F.

Question 2.90 What is A/B testing? How is it different from usual Hypothesis testing?

Question 2.91 How can you prove that one improvement you've brought to the algorithm is really an improvement over not doing anything? How familiar are you with A/B testing ?

Question 2.92 What are the metrics to evaluate a website? A search engine? Music streaming app? (Think of analogous questions regarding common metrics and intuition for things like customer engagement/retention rate, conversion, similar products, spam rate, complaint rate, ads efficiency, etc.)

Question 2.93 Most A/B testing systems use p-value to measure significance of a new feature. What is the main problem with this approach? How will you avoid it?

Question 2.94 The standard deviation of a sequence of values $x_1, x_2, \ldots, x_n$ is zero. What can you conclude from this?

Question 2.95 What is a time series?

Question 2.96 What is the difference between data for usual statistical analysis and time series data?

Question 2.97 Did you do any projects which involved dealing with time?

Question 2.98 How would you apply resampling to time series data?

Question 2.99 What is a sliding window method?

Question 2.100 Given that a population of men has normally distributed weights, with a mean of 173 lb and a standard deviation of 30 lb, find the probability that

1. If one man is randomly selected, his weight is greater than 180 lb.
2. If 36 different men are randomly selected, their mean weight is greater that 180 lb.

Question 2.101 In time series modeling, how can we deal with multiple types of seasonality like weekly and yearly seasonality?

Question 2.102 How regularly an algorithm should be updated?

Question 2.103 How does the Bayesian definition of probability differ from the frequentist definition?

Question 2.104 How a Bayesian inference differs from the frequentist version of data modeling on the same data set?

Question 2.105 Are you a Bayesian or frequentist?

Question 2.106 I want to evaluate multivariate probability density function on a high dimensional experimental data set. But the covariance matrix is non positive definite. What does this tell me about my data? What can I do about it?

Chapter 3
Machine Learning

Question 3.1 What is the difference between Data Mining and Machine Learning (ML)?

Question 3.2 Tell me about a popular application of machine learning that you see on day-to-day basis.

Question 3.3 What is your favorite ML algorithm? How will you explain it to a layman? Why is it your favorite?

Question 3.4 Discuss how would you go about feature engineering.

Question 3.5 What is the difference between supervised learning and unsupervised learning?

Question 3.6 What are your views on the relationship between ML and statistics?

Question 3.7 How Deep Learning fits (or not) in the ML field?

Question 3.8 Isn't ML just a "glorified curve fitting"?

Question 3.9 What is inductive machine learning?

Question 3.10 What are five popular ML algorithms?

Question 3.11 What is genetic programming?

Question 3.12 What are recommender systems?

Question 3.13 What is collaborative filtering?

Question 3.14 What is cosine similarity?

Question 3.15 Which libraries for data analysis do you know in Python/R/Java?

Question 3.16 Have you used *numpy, scipy, pandas,* or *scikit-learn*?

Question 3.17 What are some features of *pandas/scikit-learn* that you like? Don't like?

Question 3.18 How would you compare *TensorFlow* with *scikit-learn*?

Question 3.19 Why is "vectorization" such a powerful method for optimizing numerical code? What is going on that makes the code faster relative to alternatives like the nested *for* loop?

Question 3.20 Compare *keras, caffe, torch, TensorFlow.*

Question 3.21 Have you heard of *BLAS? LAPACK? MINUIT*?

Question 3.22 When is it better to write your own code than using a data science software package?

Question 3.23 How would you generate related searches for a search engine?

Question 3.24 How would you suggest followers on Twitter?

Question 3.25 Have you used any online platforms for machine learning such as Azure ML or AWS?

Question 3.26 What is gradient decent?

Question 3.27 Will gradient descent methods always converge to the same point?

Question 3.28 What is the difference between a convex function and non-convex?

Question 3.29 What are the two methods used for calibration in supervised learning? Which method is frequently used to prevent overfitting?

Question 3.30 What is a local minima? Is it always bad to have local minima?

Question 3.31 What is the difference between gradient descent vs. stochastic gradient descent?

Question 3.32 Is regression some type of supervised learning? Why?

Question 3.33 What are some possible problems with regression models? How do you avoid or compensate for them?

Question 3.34 What is Ordinary Least Squares (OLS) regression? How it can be learned?

Question 3.35 Can you derive the OLS regression formula for one-step solution?

Question 3.36 What is linear regression? Why is it called linear?

Question 3.37 What is the difference between linear regression and logistic regression?

Question 3.38 Name a few types of regression you are familiar with. What are the differences?

Question 3.39 How would you find a ranked list of the most significant features in regression?

Question 3.40 What are the assumptions required for linear regression?

Question 3.41 What are the constraints you need to keep in mind when using a linear regression?

Question 3.42 What if some of the linear regression assumptions are violated?

Question 3.43 Do you consider the model $Y \sim X_1 + X_2 + X_1 \cdot X_2$ to be linear? Why?

Question 3.44 How does the variance of the error term change with the number of predictors in OLS?

Question 3.45 A learning algorithm with low bias and high variance may be suitable under what circumstances?

Question 3.46 Do we always need the intercept term in regression? When do we need it and when do we not?

Question 3.47 OLS is to linear regression what maximum likelihood is to logistic regression. Explain the statement.

Question 3.48 Why do we need a bias term in linear regression?

Question 3.49 Why do we call it "general linear model" (GLM) when it's clearly nonlinear?

Question 3.50 What are some situations where a general linear model fails? What are the drawbacks of general linear model? Are

you familiar with alternatives (Lasso, ridge regression, boosted trees)?

Question 3.51 What does it practically mean for a design matrix to be *ill-conditioned*?

Question 3.52 When does regularization become necessary in machine learning?

Question 3.53 What is regularization?

Question 3.54 What is ridge regression? How is it different from OLS regression? Why do we need it?

Question 3.55 What is lasso regression? How is it different from OLS and ridge?

Question 3.56 When is ridge regression favorable over lasso regression?

Question 3.57 Lasso regression uses ℓ_1-norm of coefficients as the penalty term, while ridge regression uses the ℓ_2-norm. Which of these regularization methods is more likely to result in sparse solutions, where one or more coefficients are exactly zero?

Question 3.58 Geometrically speaking, why does lasso produce solutions with zero-valued coefficients while ridge doesn't?

Question 3.59 What is collinearity?

Question 3.60 How would you remove multi-collinearity?

Question 3.61 After analyzing the model, your manager has informed that your regression model is suffering from multi-collinearity. How would you check if he is correct? Without losing any information, can you still build a better model?

Question 3.62 What are R^2 and adjusted R^2? How they help in evaluating the performance of a regression model?

Question 3.63 You have been asked to evaluate a regression model based on R^2, adjusted R^2, and tolerance. What will be your criteria?

Question 3.64 How would you validate the predictive model you created for a quantitative outcome variable using multiple regressions.

Question 3.65 How would you check if the regression model fits the data well?

Question 3.66 In linear regression, under what condition R^2 always equals a perfect 1?

Question 3.67 What is overfitting in a regression model? What are ways to avoid it?

Question 3.68 I know that a linear regression model is generally evaluated using adjusted R^2 or F value. How would you evaluate a logistic regression model?

Question 3.69 What is logistic regression?

Question 3.70 How do we interpret the coefficients in logistic regression?

Question 3.71 What is a linear model?

Question 3.72 Is logistic regression a linear classifier? Why?

Question 3.73 Compare logistic regression with decision trees (or, neural networks).

Question 3.74 Can you describe what is the classification problem?

Question 3.75 What is the simplest classification algorithm?

Question 3.76 What classification algorithms do you know? Which one you like the most?

Question 3.77 What are the advantages of Naïve Bayes?

Question 3.78 Why is Naïve Bayes so "naïve"?

Question 3.79 Explain prior probability, likelihood, and marginal likelihood in the context of Naïve Bayes algorithm?

Question 3.80 How would you improve a spam detection algorithm that uses Naïve Bayes?

Question 3.81 How can we use Naïve Bayes classifier for categorical features?

Question 3.82 What other models do you know beside Naïve Bayes?

Question 3.83 What is Fisher Discriminant Analysis? How it is different from PCA? Is it supervised or not?

Question 3.84 What is an Artificial Neural Network (ANN)? How to train an ANN? What is backpropagation?

Question 3.85 How does a neural network with three layers (one input layer, one inner layer, and one output layer) compare to a logistic regression?

Question 3.86 What is the activation function in ANN? What are some of the functions used for this?

Question 3.87 Why do we need activation function in the neural networks training?

Question 3.88 Why the hyperbolic tangent function is the preferred activation function?

Question 3.89 What is softmax regression? Where is it used most often?

Question 3.90 How are neural networks related to Fourier transform?

Question 3.91 I'm trying to fit a single hidden layer neural network to a given dataset, and I find that the weights are oscillating a lot over training iterations — varying wildly, often swinging between positive and negative values. What parameter do I need to tune to address this issue?

Question 3.92 What is Deep Learning and what are some of the main characteristics that distinguish it from traditional ML?

Question 3.93 How are recurrent neural networks different from convolutional neural networks?

Question 3.94 What is vanishing gradient problem in a recurrent neural network? How can you avoid it?

Question 3.95 How would you solve the vanishing gradient problem?

Question 3.96 How do Long Short Term Memory (LSTM) networks avoid the vanishing gradient problem seen in recurrent neural networks?

Question 3.97 What is a kernel?

Question 3.98 How are kernel methods different? Explain the kernel trick.

Question 3.99 Tell me about positives and negatives of using Gaussian processes/kernel methods approach to learning.

Question 3.100 How does a kernel method scale with the number of instances (e.g., with a Gaussian rbf kernel)?

Question 3.101 Describe ways to overcome scaling issues in kernel methods.

Question 3.102 When training a Support Vector Machine (SVM), what value are you optimizing for?

Question 3.103 What is the maximum margin classifier? How this margin can be achieved and why is it beneficial?

Question 3.104 How do we train SVM? What about hard SVM and soft SVM?

Question 3.105 Which kernels do you know? How to choose a kernel?

Question 3.106 What is convex hull in the context of SVM?

Question 3.107 Give me some tips on practical use of SVM.

Question 3.108 What is a decision tree?

Question 3.109 What are some business reasons you might want to use a decision tree model?

Question 3.110 How do you build a decision tree model?

Question 3.111 How do you split the node? Describe some of the different splitting rules used by different decision tree algorithms.

Question 3.112 What impurity measures do you know?

Question 3.113 Is a big brushy tree always good?

Question 3.114 What is pruning and why is it important?

Question 3.115 How can one prune the tree?

Question 3.116 Why do we combine multiple trees? Is it a good idea?

Question 3.117 What is ensemble learning? When should we use it?

Question 3.118 What is Random Forest? Why is it good?

Question 3.119 What are some other ways to combine trees? What about *boosting*?

Question 3.120 What is the difference between *bagging* and *boosting*?

Question 3.121 Both being tree based algorithms, how is Random Forest different from Gradient Boosting Machine (GBM) algorithm?

Question 3.122 Running a binary classification tree algorithm is the easy part. Do you know how the tree splitting takes place, i.e., how does the tree decide which variable to split at the root node and succeeding nodes?

Question 3.123 Give me some tips on practical use of decision trees.

Question 3.124 How would you tune a random forest?

Question 3.125 What cross-validation technique would you use on a time series data set? Is it k-fold or LOOCV?

Question 3.126 What is the cluster analysis problem? Which cluster analysis methods you know?

Question 3.127 Describe *K*-means. How's it trained? What is the objective of *K*-means?

Question 3.128 How would you evaluate the quality of the clusters that are generated by a run of *K*-means?

Question 3.129 What is the computational complexity of a good, fast clustering algorithm? What is a good clustering algorithm? How do you determine the number of clusters?

Question 3.130 How do you select *K* for *K*-Means?

Question 3.131 How is kNN different from *K*-means clustering?

Question 3.132 In both *K*-means and kNN, we use Euclidean distance to calculate the distance between nearest neighbors. Why not Manhattan distance?

Question 3.133 How can you modify *K*-Means to produce soft class assignments?

Question 3.134 Describe any other cluster analysis method, e.g., DBSCAN.

Question 3.135 When should we use *k*-Nearest Neighbors for regression?

Question 3.136 Have you heard of Deep Forest? What are some of its advantages compared to Deep Neural Networks?

Question 3.137 Describe Markov models? Give an example of the Markov process.

Question 3.138 What is a Markov chain?

Question 3.139 What is a Hidden Markov model?

Question 3.140 What is a probabilistic graphical model?

Question 3.141 What is the difference between Markov networks and Bayesian networks?

Question 3.142 Have you heard of "No Free Lunch" theorem in the context of machine learning and statistical inference?

Question 3.143 Can you explain some of the extensions of linear models, e.g., Splines or Lowess?

Question 3.144 What is adversarial machine learning? What you know about the Generative Adversarial Networks (GANs)?

Question 3.145 Have you heard of CRISP-DM? Tell me about it.

Chapter 4
Data Modeling

Question 4.1 Tell me about a time when you worked on a project involving ML, or optimized an algorithm for performance, accuracy etc.

Question 4.2 Do you have experience with Scikit-learn (or Weka, R, TMVA, Spark, TensorFlow, Torch, etc.)? Tell me what you've done with that. Write some example data pipelines in that environment.

Question 4.3 Estimate the amount of time in your past project spent on each segment of your data mining/machine learning work.

Question 4.4 Tell me about an original algorithm you've created.

Question 4.5 Give examples of data cleaning techniques you have used in the past.

Question 4.6 Rise in global average temperature led to decrease in number of pirates around the world. Does that mean that decrease in number of pirates caused the climate change?

Question 4.7 Your model considers the feature X significant, and Z is not, but you expected the opposite result. How will you explain it?

Question 4.8 How would you effectively represent data with 5 dimensions?

Question 4.9 You're trying to find the best place to put in an advertisement banner on your website. You can make its size small, medium, or large and choose vertical position top, middle, or bottom. At least how many total page visits (n) and ad clicks (m) do you need to say with 95% confidence that one of the designs performs better than all other possibilities?

Question 4.10 A dairy farmer is trying to understand the factors that affect milk production of her cattle. She has been keeping logs of the daily temperature (usually $30 - 40°$ C), humidity ($60 - 90\%$), feed consumption ($2000 - 2500$ kg), and milk produced ($500 - 1000$ liters). How would you begin processing the data in order to model it, with the goal of predicting liters of milk produced in a day? What kind of machine learning problem is this?

Question 4.11 You are given a train data set having 1000 columns and 1 million rows. The data set is based on a classification problem. Your manager has asked you to reduce the dimension of this data so that model computation time can be reduced. Your machine has memory constraints. What would you do? (You are free to make practical assumptions.)

Question 4.12 Your company is building a facial expression coding system, which needs to take input images from a standard HD 1920×1080 pixel webcam, and continuously tell whether the user is in one of the following states: neutral, happy, sad, angry, or afraid. When the user's face is not visible in the camera frame, it should indicate a special state: none. What class of machine learning problems does this belong to? If each pixel is made up of 3 values (for red, green, blue channels), what is the raw input data complexity (number of dimensions) for processing each image? Is there a way to reduce the number of dimensions? How would you encode the output of the system? Explain why.

Question 4.13 Climate data collected over the past century reveals a cyclic pattern of rising and falling temperatures. How would you model this data (a sequence of average annual temperature values) to predict the average temperature over the next 5 years?

Question 4.14 Your job at an online news service is to collect text reports from around the world, and present each story as a single article with content aggregated from different sources. How would you go about designing such a system? What ML techniques would you apply?

Question 4.15 You are assigned a new project which involves helping a food delivery company save more money. The problem is, company's delivery team aren't able to deliver food on time. As a result, their customers get unhappy. And, to keep them happy, they end up delivering food for free. Which machine learning algorithm can save them?

Question 4.16 "People who bought this, also bought ..." recommendations seen on Amazon is a result of which algorithm?

Question 4.17 What is dimension reduction in Machine Learning?

Question 4.18 What is the curse of dimensionality and how should one deal with it when building machine-learning models?

Question 4.19 What are the problems of large feature space? How does it affect different models, e.g. OLS? What about computational complexity?

Question 4.20 What are eigenvalue and eigenvector?

Question 4.21 What is Principal Component Analysis (PCA)? What is the problem it solves?

Question 4.22 How is PCA related to eigenvalue decomposition (EVD)?

Question 4.23 What is Independent Component Analysis (ICA)? What's the difference between ICA and PCA?

Question 4.24 Is PCA a linear transformation or not? Why?

Question 4.25 What's the relationship between PCA and SVD? When SVD is better than EVD for PCA?

Question 4.26 You are given a data set. The data set contains many variables, some of which are highly correlated and you know about it. Your manager has asked you to run PCA. Would you remove correlated variables first? Why?

Question 4.27 Under what conditions is PCA effective?

Question 4.28 Suppose you have a very sparse matrix where rows are highly dimensional. You project these rows on a random vector of relatively small dimensionality. Is it a valid dimensionality reduction technique or not?

Question 4.29 Why do we need to center data for PCA and what can happen if we don't do it?

Question 4.30 Do we need to scale data for PCA?

Question 4.31 Is rotation necessary in PCA? If yes, Why? What will happen if you don't rotate the components?

Question 4.32 What is the difference between density-sparse data and dimensionally-sparse data?

Question 4.33 What dimensionality reductions can be used for preprocessing the data?

Question 4.34 While working on a data set, how do you select important variables? Explain your methods.

Question 4.35 What is Feature Engineering? Why do we need it?

Question 4.36 Are all features equally good?

Question 4.37 What is the downfall of using too many or too few variables?

Question 4.38 How many features should you use? How do you select the best features?

Question 4.39 How to go from categorical variables to numerical?

Question 4.40 We know that one-hot encoding increases the dimensionality of a data set, but label encoding doesn't. How?

Question 4.41 What to do with categorical variables of high cardinality?

Question 4.42 How do you handle missing data?

Question 4.43 You are given a data set. The data set has missing values which spread along 1 standard deviation from the median. What percentage of data would remain unaffected? Why?

Question 4.44 You are given a data set consisting of variables having more than 30% missing values? Let's say, out of 50 variables, 8 variables have missing values higher than 30%. How will you deal with them?

Question 4.45 Describe several feature selection methods. Are these methods depend on the model or not?

Question 4.46 We have a big data set. What is your plan for dealing with outliers?

Question 4.47 What is one way that you would handle an imbalanced dataset that's being used for prediction? (i.e., vastly more neg-

ative classes than positive classes.)

Question 4.48 How do you deal with sparsity?

Question 4.49 What are black and white lists? (In the context of spam detection)

Question 4.50 What are positive rules in the context of fraud or spam detection?

Question 4.51 You are training an image classifier with limited data. What are some ways you can augment your dataset?

Question 4.52 How would you test if survey responses were filled at random by certain individuals, as opposed to truthful selections?

Question 4.53 What is "overfitting" in machine learning? How can you avoid overfitting?

Question 4.54 Explain the "bias-variance trade-off". Why it is fundamental to machine learning?

Question 4.55 How do you know if your model overfits?

Question 4.56 Which evaluation metrics you know? Something apart from accuracy?

Question 4.57 Why is mean square error a bad measure of model performance? What would you suggest instead?

Question 4.58 You are given a data set on cancer detection. You've build a classification model and achieved an accuracy of 96%. Why shouldn't you be happy with your model performance? What can you do about it?

Question 4.59 After spending several hours, you are now anxious to build a high accuracy model. As a result, you build 5 GBM models,

thinking a boosting algorithm would do the magic. Unfortunately, neither of models could perform better than benchmark score. Finally, you decided to combine those models. Though, ensemble models are known to return high accuracy, but you are unfortunate. Where did you miss?

Question 4.60 You are working on a classification problem. For validation purposes, you've randomly sampled the training data set into train and validation. You are confident that your model will work incredibly well on unseen data since your validation accuracy is high. However, you get shocked after getting poor test accuracy. What went wrong?

Question 4.61 What precision and recall are? What is ROC curve?

Question 4.62 Suppose you wanted to keep a record of some computations that your model performs while in production. How would you go about doing this?

Question 4.63 How would you monitor that the performance of a model you trained does not degrade over time?

Question 4.64 I have two models of comparable accuracy and computational performance. Which one should I choose for production and why?

Question 4.65 What is cross-validation?

Question 4.66 What is 10-fold cross-validation?

Question 4.67 What is the difference between holding out a validation set and doing 10-fold CV?

Question 4.68 How do you know if one machine learning algorithm is better than other?

Question 4.69 You have built several different models. How would you select the best one?

Question 4.70 Explain to a business analyst the trade-off between the predictive power and the interpretability of a model — and why this matters.

Question 4.71 What is sensitivity analysis? Is it better to have low sensitivity (that is, great robustness) and low predictive power, or the other way around?

Question 4.72 What do you think about the idea of injecting noise in your data set to test the sensitivity of your models?

Question 4.73 You have one model and want to find the best set of parameters for this model. How would you do that?

Question 4.74 How would you look for the best parameters? Do you know something else apart from grid search?

Question 4.75 In your opinion, which is more important when designing a machine learning model: model performance or model accuracy?

Question 4.76 Can you estimate and forecast sales for any book, based on Amazon.com public data?

Question 4.77 You are compiling a report for user content uploaded every month and notice a spike in uploads in October. In particular, a spike in picture uploads. What might you think is the cause of this, and how would you test it?

Question 4.78 Can you perform logistic regression with Excel? How? Would the result be good?

Question 4.79 How will you make Yelp reviews more "useful"?

Question 4.80 How can you design an algorithm to estimate the number of LinkedIn connections your colleagues have? The number is readily available on LinkedIn for your first-degree connections with up to 500 LinkedIn connections. However, if the number is above 500, it just says 500+. The number of shared connections between you and any of your LinkedIn connections is always available.

Question 4.81 What are some machine learning problems encountered specifically when dealing with small data samples?

Question 4.82 How do we know if we have collected enough data to train a model?

Question 4.83 Suppose we are training a model using a particular optimization procedure such as stochastic gradient descent. How do we know if we are converging to a solution? If the training procedure converges will it always result in the best possible solution?

Question 4.84 Describe different pre-processing steps that you might carry out on data before using them to train a model. State under what conditions they might be applied.

Question 4.85 Describe the steps you follow when designing data-driven model to tackle a business problem. An example might be to automatically classify customer support emails by topic or sentiment. Another might be to predict a company's employee churn.

Question 4.86 Give examples of good and bad visualizations.

Question 4.87 What visualization tools (Tableau, D3.js, ggplot, matplotlib, and so on) have you used?

Question 4.88 How would you build a data-driven recommender system?

Question 4.89 What are the limitations of the data-driven recommender system approach?

Question 4.90 I've found some data about wheat-growing regions in Europe that include annual rainfall (R, in inches), mean altitude (A, in meters), and wheat output (O, in kg/km^2). A rough analysis and some plots make me believe that output is related to the square of rainfall, and log of altitude: $O = \beta_0 + \beta_1 R_2 + \beta_2 \log A$. Can I fit the coefficients (β's) in my model to the data using linear regression?

Question 4.91 You've got a data set to work with having p (number of variables) $> n$ (number of observations). Why is OLS a bad option here? Which techniques would be better to use? Why?

Question 4.92 You have built a regression model. Your model R^2 isn't as good as you wanted. For improvement, you remove the intercept term and your model R^2 becomes 0.8 from 0.3. Is it possible? How?

Question 4.93 Do you suggest that treating a categorical variable as continuous would result in a better predictive model?

Question 4.94 You came to know that your model is suffering from low bias and high variance. Which algorithm should you use to tackle it? Why?

Question 4.95 Considering the long list of machine learning algorithms, given a data set, how do you decide which one to use?

Question 4.96 How can you prove that the one improvement you've brought to an algorithm is really an improvement over not doing anything?

Question 4.97 When training a 10-layer neural network using backpropagation, I find that the weights for the top 3 layers are not changing at all! The next few layers ($4 - 6$) are changing, but very slowly. What's going on and how do I fix this?

Question 4.98 You've built a random forest model with 10000 trees. You got delighted after getting training error as 0.0. But, the validation

error is 34.23. What is going on? Haven't you trained your model perfectly?

Question 4.99 Do you think 50 small decision trees are better than a large one? Why?

Question 4.100 You are working on a time series data set. Your manager has asked you to build a high accuracy model. You start with the decision tree algorithm, since you know it works fairly well on all kinds of data. Later, you tried a time series regression model and got higher accuracy than decision tree model. Can this happen? Why?

Question 4.101 How would you perform clustering on one million unique keywords, assuming you have 10 million data points — each consisting of two keywords and a metric measuring how similar these two keywords are? How would you create this 10 million data points table in the first place?

Chapter 5

Natural Language Processing

Question 5.1 What is NLP? How is it related to machine learning?

Question 5.2 How would you turn unstructured text data into structured data usable for ML models?

Question 5.3 Is it necessary to turn unstructured data into structured data? Or is it OK to store data as flat text files rather than in an SQL-powered RDBMS?

Question 5.4 What are some of the models to characterize a body of text?

Question 5.5 What is a Bag of Words?

Question 5.6 What is the Vector Space Model?

Question 5.7 What is TF-IDF (Term Frequency – Inverse Document Frequency)? What is the significance of TF-IDF?

Question 5.8 How is the TF-IDF score computed?

Question 5.9 What is *word2vec*? How it can be used in NLP and information retrieval?

Question 5.10 What is *N*-Grams? How is it used in Language Models?

Question 5.11 Describe latent Dirichlet allocation (LDA) topic modeling.

Question 5.12 What's the disadvantage of LDA for short texts?

Question 5.13 What is latent semantic indexing (LSI) and where can it be applied?

Question 5.14 What is the difference between LDA and LSI?

Question 5.15 How would you train a model that identifies whether the word "Apple" in a tweet belongs to the fruit or the company?

Question 5.16 How would you find all the occurrences of quoted text in a news article?

Question 5.17 How would you build a system that auto corrects text that has been generated by a speech recognition system?

Question 5.18 How would you build a system to translate English text to Greek and vice-versa?

Question 5.19 How would you build a system that automatically groups news articles by subject?

Question 5.20 What are stop words? Why do we remove stop words? When do we not remove them? Describe an application in which stop words shouldn't be removed.

Question 5.21 How would you design a model to predict whether a movie review was positive or negative?

Question 5.22 What is entropy?

Question 5.23 How would you estimate the entropy of the English language?

Question 5.24 How does the PageRank algorithm work?

Question 5.25 What is dependency parsing? What is the difference between constituency parsing and dependency parsing?

Question 5.26 What is constituency parsing? Can a constituency tree be converted to dependency tree? What about vice versa?

Question 5.27 What is the difference between shallow parsing and deep parsing? How are these connected to dependency parsing?

Question 5.28 What are the difficulties in building and using an annotated corpus of texts, such as the Brown corpus and what can be done to mitigate them?

Question 5.29 How will you cluster short text tweets? What problems you expect to face during this process?

Question 5.30 What are the trade-offs between supervised and unsupervised methods specific to NLP problems like word sense disambiguation or machine translation etc.?

Question 5.31 How would you approach a problem in NLP that is very easily solved for English (where you have abundant resources like WordNet, Dictionaries, Sense tagged and parallel corpora) for other resource deprived languages like Mandarin, Arabic, etc.?

Question 5.32 What are the linguistic properties that are invariant across languages?

Question 5.33 Which is better to use while extracting features — character n-grams or word n-grams? Why?

Question 5.34 What is a parts-of-speech (POS) tagger? How can you build one?

Question 5.35 Which is a better algorithm for POS tagging — SVM or hidden Markov models? Why? How would you deal with unknown words?

Question 5.36 What packages are you aware of in Python that are used in NLP and ML?

Question 5.37 How can you optimize a web crawler to run much faster, extract better information, and better summarize data to produce cleaner databases?

Question 5.38 What is probabilistic merging (a.k.a. fuzzy merging)? Is it easier to handle with SQL or other languages? Which languages would you choose for semi-structured text data reconciliation?

Question 5.39 Why *K*-means clustering is not very effective in clustering similar stories (or topics) together in streaming data? What would you do instead?

Question 5.40 What is the skip-gram model?

Question 5.41 What is lexical ambiguity?

Question 5.42 What is syntax level ambiguity?

Question 5.43 What are some of the steps involved in preprocessing messy real world text data?

Question 5.44 What is fuzzy matching of records?

Question 5.45 Tell me about some commonly used stemmers. How are they different from one another?

Question 5.46 Explain how a lemmatizer works. Why we need a lexical database for the lemmatizer to work well?

Question 5.47 How would you measure similarity (or, distance) between two strings?

Question 5.48 What is Levenshtein distance?

Question 5.49 What is Jaro distance? How is it different from Jaro-Winkler distance?

Question 5.50 What are some typical use cases for the Levenshtein and Jaro distance metrics?

Question 5.51 What is Hamming distance? What are some of its applications?

Question 5.52 What is Jaccard similarity? How it can be used to detect plagiarism?

Question 5.53 What is referential ambiguity?

Question 5.54 What is semantic analysis?

Question 5.55 What is syntactic analysis?

Question 5.56 In your experience what are some of the challenges involved in the de-duplication and cleaning of text data records?

Chapter 6

Distributed Systems & Big Data

Question 6.1 What is the biggest data set that you have processed and how did you process it? What was the result?

Question 6.2 What are some tools for parallelizing machine learning algorithms?

Question 6.3 What is ACID?

Question 6.4 What is CAP theorem?

Question 6.5 What are some commonly used methods to resolve concurrency control?

Question 6.6 An SQL table contains two columns: student and grade. How would you find all students whose grades are above the average grade?

Question 6.7 In an SQL database, how would you find all employees who are also managers?

Question 6.8 Write an SQL query to find the second highest salary in the employee database.

Question 6.9 Write an SQL query to find the number of employees ordered by gender whose DOB is between 01/01/1960 to 12/31/1975.

Question 6.10 Write an SQL query to find duplicate rows in a database.

Question 6.11 What is the purpose of the group functions in SQL? Give some examples of group functions.

Question 6.12 Tell me the difference between an inner join, left join/right join, and union.

Question 6.13 What does UNION do? What is the difference between UNION and UNION ALL?

Question 6.14 What is the difference between SQL and MySQL or SQL Server?

Question 6.15 If a table contains duplicate rows, does a query result display the duplicate values by default? How can you eliminate duplicate rows from a query result?

Question 6.16 Explain star schema.

Question 6.17 What are examples of "embarrassingly parallelizable" algorithms?

Question 6.18 What are the basic differences between relational database and HDFS?

Question 6.19 Describe a NoSQL technology you're familiar with. What it is good at and what it is bad at?

Question 6.20 What is RDD?

Question 6.21 Have you used Hadoop, Spark, Flink, Mahout? In what context?

Question 6.22 Discuss map-reduce or your favorite parallelization abstraction. Why is map-reduce referred to as a "shared-nothing" architecture (clearly the nodes have to share something, no?)? What are the advantages/disadvantages of "shared-nothing"?

Question 6.23 What are some examples in which map-reduce works well?

Question 6.24 What are some examples in which map-reduce does not work well?

Question 6.25 Can you implement word count in map-reduce? What about something a bit more complex like TF-IDF? Naïve Bayes?

Question 6.26 What is load balance? How to make sure a map-reduce application has good load balance?

Question 6.27 In the map-reduce paradigm, what does the map function do and what does the reduce function do? What do the combiner and partitioner do?

Question 6.28 Pick an algorithm. How would you implement its parallel version? (You may be asked to write pseudo-code.)

Question 6.29 How would you estimate the median of a dataset that is too big to hold in memory?

Question 6.30 What are the trade-offs between closed form and iterative implementations of an algorithm, in the context of distributed systems?

Question 6.31 What are the security issues involved with the cloud?

Question 6.32 What is a Sparse Vector?

Question 6.33 What are the key features of Apache Spark that you like?

Question 6.34 How Spark uses Hadoop?

Question 6.35 Describe common workflow of a Spark program.

Question 6.36 What are some of the disadvantages of Spark?

Question 6.37 Hadoop uses replication to achieve fault tolerance. How is this achieved in Spark?

Question 6.38 What is lineage graph?

Question 6.39 What are the Partitions (in the context of RDD)?

Question 6.40 What is the difference between *persist*() and *cache*()?

Question 6.41 Explain what are transformations and actions in the context of RDDs.

Question 6.42 What do you understand by Lazy Evaluation (in the context of Spark)?

Question 6.43 Can RDD be shared between SparkContexts?

Question 6.44 How RDDs can be created using SparkContext? Give a few examples.

Question 6.45 How RDD helps parallel job processing?

Question 6.46 What is a DataFrame?

Question 6.47 What is Data locality?

Question 6.48 What kind of data sources Spark can process?

Question 6.49 What is Spark Streaming?

Question 6.50 Name some sources from where Spark streaming component can process real-time data.

Question 6.51 What are some cluster managers in Spark? Which one should be preferable?

Question 6.52 What are the core components of a distributed Spark application?

Question 6.53 What is the default level of parallelism in Spark?

Question 6.54 How many concurrent tasks Spark can run for an RDD partition?

Question 6.55 Please describe how execution starts and ends on RDDs in a Spark job.

Question 6.56 What limits the maximum size of a partition?

Question 6.57 What is Shuffling?

Question 6.58 How can you remove the elements with a key present in any other RDD?

Question 6.59 Why *reduceByKey* is preferable over *roupByKey*?

Question 6.60 When you call join operation on two pair RDDs, e.g., (K, V) and (K, W), what is the result?

Question 6.61 What are some commonly used libraries based on the Spark ecosystem?

Question 6.62 We have a text file possibly containing some duplicate lines. Write a Spark query to count how many times each line of text occurs in a file.

Question 6.63 Write a Spark query to count the number of words in a text file located in HDFS and store the result in a new text file.

Question 6.64 Write a Spark query to search through the "ERROR: DivisionByZero" error messages in a log file.

Question 6.65 I have a simple MySQL table called "people", which has two columns, *name* and *age*. Write a Spark query to read this table and calculate the number of people for every age. Then, save the calculated result to S3 in the JSON format.

Question 6.66 How would you compute the area of an arbitrary closed curve that is bounded in the region $[0, a]$ on the X-axis and $[0, b]$ on the Y-axis?

Question 6.67 Write a Spark query to compute the value of π using random numbers.

Question 6.68 What is a graph database? How is it different from SQL and NoSQL? What are some commonly used graph databases?

Question 6.69 What is your definition of Big Data?

Chapter 7

Culture-fit Questions

Question 7.1 Summarize your experience.

Question 7.2 What do you think makes a good data scientist?

Question 7.3 Which data scientists do you admire most and why?

Question 7.4 Which company do you admire most and why?

Question 7.5 What companies you have worked at? What was your role?

Question 7.6 What projects you implemented? Discuss some of them in detail.

Question 7.7 What are your favorite data science websites?

Question 7.8 What is the biggest data set you ever processed and how did you process it? What was the result?

Question 7.9 Tell me two success stories about your analytic projects. How was the success measured?

Question 7.10 Have you taken any data-science related online courses? If yes, how many did you complete with a certificate?

Question 7.11 Have you participated in any data science challenges? If yes, can you describe one of them?

Question 7.12 Give a few examples of "best practices" in data science.

Question 7.13 Can you outline the steps in a data science project?

Question 7.14 How do you stay up-to-date and keep your skills sharp?

Question 7.15 What/when is the latest data science book/article you read?

Question 7.16 What/when is the latest data mining conference/-workshop you attended?

Question 7.17 Do you believe in perfection or excellence?

Question 7.18 Tell me about a time when you took initiative.

Question 7.19 How well do you adapt to change?

Question 7.20 Are you a lone coder, a production guy, or an architect by nature?

Question 7.21 What are you looking for in your next team? What is important to you culturally?

Question 7.22 What are your top 5 predictions for the next 20 years?

Question 7.23 What is teamwork to you?

Question 7.24 Who are the best people you recruited and where are they today?

Question 7.25 What was the greatest work day of your life?

Question 7.26 Tell me about a time where you had to overcome a dilemma.

Question 7.27 Tell me about a time where you resolved a conflict.

Question 7.28 Tell me about a time you failed. What you have learned from it?

Question 7.29 What was the best way you delegated a task?

Question 7.30 Tell me about *<a job on your resume>*. Why did you choose to do it and what do you like most about it?

Question 7.31 Tell me about a challenge you have overcome while working on a group project.

Question 7.32 When you encounter a tedious, boring task, how would you deal with it and motivate yourself to complete it?

Question 7.33 What have you done in your previous job that you are really proud of?

Question 7.34 Tell me about a project you worked on that succeeded in part because of the way results were communicated. What were the factors that made it a success?

Question 7.35 Tell me a compelling story about data that you have analyzed.

Question 7.36 What's a project you would want to work on at our company?

Question 7.37 What unique skills do you think you'd bring to the team?

Question 7.38 What data would you love to acquire if there were no limitations?

Question 7.39 In your opinion, what are the three most important qualities of an ideal job?

Question 7.40 What personal projects do you have going on outside of the work?

Question 7.41 Have you ever thought about creating a startup? Around which idea?

Question 7.42 What can your hobbies tell me that your resume can't?

Question 7.43 What did you do today? (Or, what did you do this week?)

Question 7.44 What was the time you didn't know how to do something?

Question 7.45 If you won a million dollars in lottery, what would you do with the money?

Question 7.46 What is one thing you believe that most people do not?

Question 7.47 What personality traits do you butt heads with?

Question 7.48 What are you passionate about?

Question 7.49 What is your career plan?

Question 7.50 Do you prefer to work for a company that is in build mode or maintenance mode?

Question 7.51 What coding style do you use?

Question 7.52 Tell me about the first job you've ever had. What did you do to learn the ropes?

Question 7.53 Describe a time when your team or company was undergoing some change. How did that impact you, and how did you adapt?

Question 7.54 Describe a situation in which you were able to use persuasion to successfully convince your team leader to see things your way.

Question 7.55 Tell me about a time when you worked on multiple projects and you were required to prioritize your tasks.

Question 7.56 Please describe an instance where you had to make a decision without all of the necessary information.

Question 7.57 Talk about a time when you had to work closely with someone whose personality was very different from yours.

Question 7.58 Describe a time when you struggled to build a relationship with someone important. How did you eventually overcome that?

Question 7.59 Give me an example of a time when something you tried to accomplish failed.

Question 7.60 Tell me about a time you needed to get information from someone who wasn't very responsive. What did you do?

Question 7.61 Describe a time when you saw some problem and took the initiative to correct it rather than waiting for someone else to

do it.

Question 7.62 Tell me about a time you were dissatisfied in your work. What could have been done to make it better?

Question 7.63 Tell me about a time when you worked under close supervision or extremely loose supervision. How did you handle that?

Question 7.64 What did you do the last time things didn't go according to plan?

Question 7.65 Tell me about a difficult decision you've made in recent years.

Question 7.66 Describe a decision you made that wasn't popular and how you handled implementing it.

Question 7.67 Did you ever postpone making a decision? Why?

Question 7.68 Describe a situation in which a detail you thought to be unimportant turned out to be very important.

Question 7.69 We all make mistakes we wish we could take back. Tell me about a time you wish you'd handled a situation differently with a colleague.

Question 7.70 What do you do if you disagree with your boss?

Question 7.71 What do you do when your schedule is interrupted? Give an example of how you handle it.

Question 7.72 Give me a specific example of a time when you used good judgment and logic in solving a problem.

Question 7.73 Give me an example of a time you were able to be creative with your work. What was exciting or difficult about it?

Question 7.74 Have you had to convince a team to work on a project they weren't thrilled about? How did you do it?

Question 7.75 Give me an example of a time when you set a goal and were able to meet or achieve it.

Question 7.76 Are you comfortable speaking in public? Have you ever presented a technical topic to a large audience?

Question 7.77 Do you have a presentation you can show us, such as on *SlideShare*?

Question 7.78 Do you have experience presenting reports and findings directly to senior management in your previous roles?

Question 7.79 Which other places you have interviewed at? When is the earliest time you can start in the new job?

Question 7.80 What is the most surprising question you have faced either in a job interview or in outreach Q&A sessions?

Appendices

Appendix A

Computer Science Fundamentals Answers

This appendix contains answers to the questions posed in Chapter 1.

Answer 1.1: We can go through the string from beginning to end. We keep a running total starting at 0. Each time we reach a new digit, we multiply the total by 10 and add the new digit. When we reach the end, we return the current total, or, if there is a negative sign, the negative of the number.

Here is a C++ code to do this

```
int atoi(const std::string& str) {
  // This is our running total
  int n = 0;

  //Let's read off the sign
  int sign = 1, start_index = 0;
  char init_char = str.at(0);
  if (init_char == '+' || init_char == '-' ) {
      start_index = 1;
      if (init_char == '-' ) sign = -1;
  }

  //Process the string beginning to end
  for (int i = start_index; i < str.size(); i++) {
      n *= 10;
      n += str.at(i) - '0';
  }

```

```
  //Finally, return the number with correct sign
  return n * sign;
}
```

Data Structure	Access	Search	Insert/ delete
Array	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Stack	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Queue	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Linked list	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Doubly-linked list	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Hash table	N/A	$\mathcal{O}(1)$ [1]	$\mathcal{O}(1)$ [1]
Binary search tree	$\mathcal{O}(\log(n))$ [1]	$\mathcal{O}(\log(n))$ [1]	$\mathcal{O}(\log(n))$ [1]

Table A.1: Average time complexity for common data structures. Space complexity for each is $\mathcal{O}(n)$.

[1] Worst case $\mathcal{O}(n)$.

Answer 1.2: Here is a C++ program to do this.

```
int fib(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fib(n-1)+fib(n-2);
}
```

Since the function calls itself twice each time it is invoked, the time complexity is $\mathcal{O}(2^n)$.

Answer 1.3: Here is a C++ program to do this.

```
#include <iostream>
#include <ctime>

int main() {
   std::srand(time(0));
   int frequency[6]={0};

   for(int i=0; i<6000; i++)  frequency[rand()\%6]++;
   for(int i=0;i<6;i++) {
      std::cout << i+1 << " was rolled " << frequency[i] << " times." << std::endl;
   }
   return 0;
}
```

Answer 1.4: We can do this in two steps:

1. Go through the string looking for spaces and reverse the individual words, e.g.,

   i like this program very much $\rightarrow$ *i ekil siht margorp yrev hcum*

2. Reverse the whole string from start to end, e.g.,

   i ekil siht margorp yrev hcum $\rightarrow$ *much very program this like i*

Answer 1.5: We do not need a fancy algorithm here. Starting from 1, we can iterate through N and successively append 0 and 1 to generate the next two binary numbers. For example, for $N = 10$, the binary numbers are

$$1101110010111011110001001l010$$

A queue would be an appropriate data structure to use in this case. Here is a C++ code to do this

```
#include <iostream>
#include <queue>

int main() {
    int N = 10;

    //Create a queue with the first element 1
    std::queue<std::string> q;
    q.push(''1'');

    //Iterate through N
    for (int i=1; i <= N; i++) {
        // Print the front element
        std::cout << q.front() << '' '';

        //Add two new elements to the queue by appending 0, 1 to the front element
        q.push(q.front() + ''0'');
        q.push(q.front() + ''1'');

        //Deque the front element
        q.pop();
    }
    std::cout << std::endl;
}
```

Both time and space complexity for this solution is $\mathcal{O}(n)$.

Answer 1.6: We can represent the arrival and departure times in minutes as integers. Since the 24-hour clock runs from minute 0 to minute

1339, the elements in either array can have at most 1340 distinct values. So, we can create a hash-map of the occupation frequency for each of these as keys. (Actually we need to compute frequency for fewer keys — those between the first arrival time and the last departure time). We loop over the *Arrival* array and for each entry we find the corresponding entry in the *Departure* array, and increment frequency value for all keys between the two entries by 1. The maximum value in the frequency map would be the minimum number of gates needed to avoid waiting. Time complexity of this solution is $\mathcal{O}(n)$ for max $n = 1340$, since the hash-map access is $\mathcal{O}(1)$.

Answer 1.7: This problem can be solved by using dynamic programming. Starting with the last digit, we can decode it and solve for the remaining $n-1$ digits. If the first two digits form a valid character in the range 1 through 26, then we decode and solve for the remaining $n-2$ digits. So, the recursion relation is

$$\text{Solution}[n] = \text{Solution}[n-1] + \text{Solution}[n-2], \qquad \text{(A.1)}$$

and the time complexity is $\mathcal{O}(2^n)$. Here is a C++ program to do the decoding.

```
#include <iostream>
#include <cstdlib>
#include <string>
#include <map>
#include <vector>

// Define the code map
std::map<int, char> CodeMap = {
  {0, '0'}, {1, 'A'}, {2, 'B'}, {3, 'C'}, {4, 'D'}, {5, 'E'},
  {6, 'F'}, {7, 'G'}, {8, 'H'}, {9, 'I'}, {10, 'J'}, {11, 'K'},
  {12, 'L'}, {13, 'M'}, {14, 'N'}, {15, 'O'}, {16, 'P'},
  {17, 'Q'}, {18, 'R'}, {19, 'S'}, {20, 'T'}, {21, 'U'},
  {22, 'V'}, {23, 'W'}, {24, 'X'}, {25, 'Y'}, {26, 'Z'}};

// Decode function
void decode(std::string code, std::string sum, std::vector<std::string>& result) {
  size_t length = code.size();

  //If the code is empty, then append decoded value to the result
  if (length==0) {
    result.push_back(sum);
    return;
  }
```

```
  //Decode the last digit of the code and call dynamic programming
  auto digit = std::atoi(code.substr(length - 1).c_str());
  decode(code.substr(0,length-1), std::string(1,CodeMap[digit])+sum, result);

  //Decode the last two digits of the code and call dynamic programming
  if (length < 2) return;
  digit = std::atoi(code.substr(length - 2).c_str());
  if (digit >=10 && digit <= 26) {
     decode(code.substr(0,length-2), std::string(1,CodeMap[digit])+sum, result);
  }
}

int main() {
   std::string code = "123";
   std::string sum = "";
   std::vector<std::string> result;
   decode(code, sum, result);

   for (const auto& x: result) {
      std::cout << x << std::endl;
   }
   return 0;
}
```

Answer 1.8: We can create a hash-map of the words and their frequencies. This would have time complexity $\mathcal{O}(n)$. Then we can use binary search tree to find find k top ranked words in $\mathcal{O}(n \log n)$ run time.

Answer 1.9: Big-endian and little-endian are terms that describe the order in which a sequence of bytes are stored in computer's memory. Big-endian is an order in which the most significant value is stored first, i.e., at the lowest storage address. In little-endian, the least significant value in the sequence is stored first. Some well-known big-endian architectures include IBM mainframes, most RISC-based computers, Motorola microprocessors, and TCP/IP addresses. On the other hand, Intel x86 processors are little-endian.

Answer 1.10: Left shifting takes a number, adds a certain number of zeros to the end, and removes the same number of bits from the beginning. For example,

$$00101011 << 4 \;=\; 10110000 \tag{A.2}$$

Likewise, right shifting takes a number, adds a certain number of copies of the first bit to the beginning, and removes the same num-

ber of bits from the end. For example,

$$10110010 >> 4 \;=\; 11111011 \tag{A.3}$$

Answer 1.11: The binary equivalent of a negative number is equal to the two's complement of the absolute value of the number. Binary equivalent of 5 is 0101. Its two's complement is $1010 + 1 = 1011$. Hence -5 in binary is 1011.

Answer 1.12: We can quickly compute 2^x by left-shifting 1 by x:

$$2^x \;=\; 1 << x \tag{A.4}$$

Answer 1.13: If a number x is a power of 2, then the logical AND of x and $x - 1$ would be zero. So, we can check whether

$$x \;\&\; (x-1) = 0 \tag{A.5}$$

Answer 1.14: This is easy!

$$15\hat{}18\hat{}15 = 18, \tag{A.6}$$

because two identical numbers in an expression involving XOR cancel out.

Answer 1.15: This is a well-known problem, which is often very efficiently implemented natively in the compilers. Below is a simple implementation in C++ that checks every bit, so it has a time complexity equal to the number of 1's, i.e., worst case $2^N - 1$ for an N-digit binary number.

```
int count_ones(uint64_t x) {
    int count;
    for (count=0; x; count++)
        x &= x-1;
    return count;
}
```

Answer 1.16: We need to first check that the least significant digit is a zero. Then we left shift by 1 and increment our count. Repeat until we find a non-zero digit. Here is a C++ implementation.

```
unsigned trailing_zeros(unsigned n) {
    if (n == 0) {
        return -1;
    }
    unsigned count = 0;
    while (n \% 2 == 0) {
        count++;
        n >>= 1;
    }
    return count;
}
```

Answer 1.17: We can represent the set as a binary number. A value equal to 1 in the n^{th} position would indicate that "object number n is in this set," while a 0 would indicate that "object number n is not in the set." This way we can form 2^n subsets.

The answer, 2^n, is also easy to arrive at using the principle of mathematical induction. Clearly, an empty set has only itself as a subset. A set with 1 element has 2^1 subsets, itself and the empty set. Let's assume that a set with n elements has 2^n subsets. Now, if we add a new element to this set, all previous subsets will still remain valid subsets. In addition, each previous subset with the new element added is also a subset. Thus, the new set with $n + 1$ elements has $2 \cdot 2^n = 2^{n+1}$ subsets.

Answer 1.18: This is a typical starter or segue question. Just describe how you ended up using your favorite programming language.

Answer 1.19: A classic starter question which generally leads to subsequent questioning. To be effective you can prepare an elevator pitch version of your favorite data science project and emphasize the programming component.

Answer 1.20: We can use hash table. We go through the array starting from the first element and insert the elements into the hash table. If the element already exists in the hash table, then we found the duplicate. Both time and space complexity in this case are $\mathcal{O}(n)$.

Please make sure to review algorithm complexity, including big-O notation. Table A.1 summarizes average time and space complexities for some of the most common data structure operations. For more information on algorithms, see Refs. [1–6].

Answer 1.21: Polymorphism is the ability of a function to have different behavior depending on the type of object it is being called on. For example, the function *Shape::Draw()* would work differently under the hood for circle than for rectangle.

Answer 1.22: A function's being *virtual* describes its behavior in an inherited class[7]. Let's say we have the following C++ code, where class *B* inherits from class *A* and both have a function named *foo()*.

```
A *a = new B();
a->foo();
```

If *foo()* is declared virtual in class *A*, then the function found in class *B* will be run. Otherwise the code will run the function from class *A*.

Answer 1.23: This is one of the classical data structure interviews questions! The main difference is that the stack is LIFO (Last In First Out), while the queue is an FIFO (First In First Out) data structure.

Answer 1.24: It takes constant time to add or remove a node in the linked list, and $\mathcal{O}(n)$ time to look up an element if we don't already have the pointer to that node.

Answer 1.25: We can keep track of two pointers in the linked list, and start them both at the beginning of the linked list. At each iteration, we advance the first pointer by one node and the second pointer by two nodes. If the two pointers are ever the same, then there is a cycle. If a pointer reaches the end of the linked list before that, then there is no cycle. This takes $\mathcal{O}(n)$ time.

Alternative answer: For every node we encounter while going through the list one by one, we can put a pointer to that node into a $\mathcal{O}(1)$-lookup time data structure, such as a hash table. Then, when we encounter a new node, we check if the pointer to that node already

exists in our hash table. This should take $\mathcal{O}(n)$ time, but also takes $\mathcal{O}(n)$ space.

Answer 1.26: Let's assume the two elements are p and q. Starting from p we move upward in the tree and store each encountered element in set-1. Similarly, starting from q we move upward and store each encountered element in set-2. We repeat this process iteratively. The first element that is added to one set when it is already present in the other set is the lowest common ancestor. This algorithm takes $\mathcal{O}(n)$ time, where n is the length of the path. To improve the solution, we actually only need to use one set instead of two.

Answer 1.27: There is exactly one path from the starting node to the lowest common ancestor (LCA) of the nodes. Similarly, there is exactly one path from the LCA to the second node. Therefore, we first find the LCA using the procedure described above. Then we just concatenate the two paths: $p \rightarrow$ LCA and LCA $\rightarrow q$.

Algorithm	Time complexity	Space complexity
Quicksort	$\mathcal{O}(n \log(n))$ [1]	$\mathcal{O}(\log(n))$
Mergesort	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n)$
Heapsort	$\mathcal{O}(n \log(n))$	$\mathcal{O}(1)$
Bubble sort	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$
Insertion sort	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$
Selection sort	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$
Tree sort	$\mathcal{O}(n \log(n))$ [1]	$\mathcal{O}(n)$
Bucket sort	$\mathcal{O}(n + k)$ [1]	$\mathcal{O}(n)$

Table A.2: Average complexity for sorting.

[1] Worst case $\mathcal{O}(n^2)$.

Answer 1.28: A binary search tree is a data structure that keeps items in sorted order. Each node's element must be greater than every element in its left subtree and less than every element in its right subtree. The height of a well-balanced binary search tree is $\mathcal{O}(\log n)$, where n is the number of elements in the tree. Insertion, removal, and lookup take $\mathcal{O}(\log n)$ time.

Answer 1.29: The second largest element is the second element in the

reverse in-order traversal. So we can traverse the given Binary Search Tree in reverse of in-order and take the second node.

Answer 1.30: The time complexity is $\mathcal{O}(n \log n)$ in the worst case and $\mathcal{O}(\log n!)$ on average. To prove this, we cannot assume the sorting algorithm is going to necessarily choose a pivot as in quicksort, or split the input as in mergesort — we need to analyze any possible comparison logic. The sorting algorithm must output a permutation of the input $[a_1, a_2, ..., a_n]$. For each of the $n!$ possible permutations, there exists an input for which that permutation is the only correct answer. So, the complexity is

$$\log_2(n!) = \log_2(n) + \log_2(n-1) + ... + \log_2(2) = \mathcal{O}(n \log n) \quad \text{(A.7)}$$

Table A.2 lists complexity for some common sorting algorithms.

Answer 1.31: Mergesort is a recursive way to sort an array. We divide the array in two, recursively sort each half, and then combine them into a sorted array. The algorithm relies on the fact that one can quickly combine two sorted arrays into a single sorted array. One way to do so would be by keeping two pointers into the two sorted arrays. We repeatedly insert the smaller of the two values pointed to by the pointers to the combined array and advance the pointer. The algorithm takes $\mathcal{O}(n \log n)$ time — both on average and in the worst case.

Answer 1.32: Quicksort is another sorting algorithm. Here we first selects a random element of the array as the "pivot". Then divides the array into two groups — one with elements less than the pivot and the other with elements greater than the pivot. Then, we recursively sort both groups. Quicksort also has expected time complexity of $\mathcal{O}(n \log n)$ and is quite fast in practice. However, it has $\mathcal{O}(n^2)$ worst-case running time. So, if we need guaranteed $\mathcal{O}(n \log n)$ performance, we should use mergesort instead.

Answer 1.33: Using quicksort or mergesort.

Answer 1.34: We should sort the array in ascending order of the people's number. For each person, we check the three people immedi-

ately before and after them. Their three friends will be among these six people. The algorithm will take $\mathcal{O}(n \log n)$ time.

Answer 1.35: We first select a random pivot and partition the array as in the quicksort algorithm. Then, based on the index of the pivot, we know which half the desired element lies in. To identify the the k^{th} smallest element, we partition the second half of the array and continue recursively. The expected running time is

$$n + (n/2) + (n/4) + (n/8) + ... = \mathcal{O}(n) \tag{A.8}$$

Answer 1.36: Finding the median of an array is a special case of the previous problem, where $k = n/2$.

Answer 1.37: We can use recursion here. The top of the sorted stack should be the largest element. If the stack is non-empty, the function *sort()* pops the first element from the stack, then sorts the rest of the stack recursively, and finally inserts the first element back into the sorted stack in the right place by calling the function *insert()*. The function *insert()* takes a sorted stack and an element to be inserted. It inserts the element into the stack such that the stack is still in sorted order.

If the sorted stack is empty or if the element to be inserted is greater than the top of the sorted stack, then we just push the element onto the stack, otherwise we pop the first element from the sorted stack, recursively insert the element to the remaining sorted stack, and then push the first element back onto the sorted stack.

Alternatively, we can pop all elements from the stack into an array, sort the array, then push the array back onto the stack.

Answer 1.38: We can use binary search. We first compare the number in the middle of the array with x. If it is equal, we are done. If the number is greater, we look in the second half of the array else we look in the first half. Then we repeat the search on the appropriate half,

once again narrowing by a factor of 2. We repeat until we find x. This algorithm takes $\mathcal{O}(\log n)$ time.

Answer 1.39: For normal weighted graphs, we can use Dijkstra's algorithm to solve the problem in time $\mathcal{O}(m + n \log n)$, where n is the number of nodes and m the number of edges. If negative weights exist, then we can use Bellman-Ford algorithm, which has a time complexity of $\mathcal{O}(mn)$.

Answer 1.40: An efficient way would be to iterate through the string looking separately for even and odd length palindromes. To find odd-length palindromes, we iterate across the potential middle character in both directions until we cannot get any further (next left and right characters don't match). Similarly, to find even-length palindromes, we iterate across a pair of identical characters in both directions. The time complexity would be $\mathcal{O}(n)$ for sparse palindromes and $\mathcal{O}(n^2)$ for dense palindromes.

Answer 1.41: You surely have written code in one or more of these languages. The optimizations you made could be context dependent, but would generally result in a better execution speed. Start with a brief overview and go deeper if the interviewer is interested.

Answer 1.42: The *VTABLE* is a table of function pointers. It is maintained per class. The *VPTR* is a pointer to *VTABLE*. It is maintained per object by the compiler.

Answer 1.43: This is a small variation from the programming experience question, but a good point to showcase your bug-hunting skills!

Answer 1.44: This is a trap question because the best trick is to actually not compute the inverse of an $N \times N$ matrix! Solving equation $Ax = b$ is faster than finding A^{-1}, so we are better off solving this equation instead. The linear solver has a time complexity $\mathcal{O}(N^2)$, while the matrix inversion takes $\mathcal{O}(N^3)$.

Similarly, if we need to solve $Ax = b$ for a lot of different b's, we can factor A and save that factorization. The one-time factorization step

will cost $\mathcal{O}(N^3)$, but afterward we can solve $Ax = b$ in $\mathcal{O}(N^2)$ time for each b.

Answer 1.45: Hashing is a way to efficiently map keys with values in an associative data structure. Each key is fed to a *hash()* function, which takes some or all of its information and digests it into a single integer. The hash table consists of an array of hash buckets. To add a key-value pair to a hash table, we compute the key's hash code then insert the pair to the hash bucket in which the mapping belongs. To lookup the value for a given key, we again compute the key's hash and if present return the value stored in that bucket.

Insertion, removal, and lookup take expected $\mathcal{O}(1)$ time. In the worst-case, each key hashes to the same bucket, so each operation takes $\mathcal{O}(n)$ time.

We use hashing when we need to optimize the map for fast lookup, fast insertion, and fast removal. For example, caching student admission records in a program. However, hash tables are generally not good at iteration and sorting, so if we need a map with sorted keys then we should not use hashing.

Answer 1.46: Ideally, the hash function will assign each key to a unique bucket, but sometimes it is possible that two keys will generate an identical hash causing both keys to point to the same bucket. This is known as hash collisions.

One way to deal with collisions is to store a linked list of key-value pairs for each bucket. How frequently a collision occurs depends on both hash function and the input data, but is generally small for sufficiently random hash functions. The best way to avoid collision is to use a good hash function that distributes elements uniformly over the hash table. There are also various collision resolution techniques like double hashing.

Answer 1.47: It is better to have one big hash table. The reason is that the hash tables have $\mathcal{O}(1)$ average lookup time and $\mathcal{O}(n)$ space requirement. So, the lookup time doesn't depend on size. $\mathcal{O}(n)$ space

implies that we don't gain anything from breaking up the hash table into smaller pieces. All being equal, it is easier and more straightforward to deal with just one table.

Answer 1.48: We can use a hash table. In the pre-computing step, we go through each word in the dictionary, sort the letters of the word in alphabetical order (so *apple* would become *aelpp*), and add the sorted letters as a key in the table and the original word as one of the values for that key. During the test run, we'll sort the input string and look up the value in the hash table. The running time will be $\mathcal{O}(n \log n)$ for sorting the string and $\mathcal{O}(1)$ for the lookup in the hash table.

Answer 1.49: This is a small variation from the previous question. As a preprocessing step, we can convert all dictionary words into multiple undirected graphs. Each graph contains words of a particular size (2, 3, 4, ..., etc.). If two graph nodes have edit distance 1, then we connect them by an edge. Once we have this graph, the problem is straightforward: Find the shortest path from a given node to another node. We can do this using the breadth-first search.

Answer 1.50: It's a program in which multiple threads in a single process have access to the same memory. The processor loads and saves a separate set of registers for each thread. The process can have one or more threads and the processor switches between the threads.

Answer 1.51: Mutex is a lock to ensure that only one thread can access a shared resource at a time. For example, while one thread is modifying an array, the processor may switch to another thread. The second thread would then have access to a partially modified version of the array. To prevent this, we could use a mutex.

A mutex is an integer that starts at 1. Whenever a thread needs to alter the array, it locks the mutex by decreasing the value by 1. This causes the thread to wait until the number is positive. When the thread is done modifying, it unlocks the mutex by incrementing the value by 1. This ensures that no two threads will modify the array at the same time.

Answer 1.52: Semaphore is also a mechanism to prevent multiple threads simultaneously modifying a shared resource. It is more general than mutex and may start at a number greater than 1. The number at which a semaphore starts is the number of threads that may access the resource at once. Semaphores support *wait()* and *signal()* operations, which are analogous to the *lock()* and *unlock()* operations of mutexes.

Answer 1.53: Deadlock is a situation where two threads have both locked the access to a shared memory and are waiting for each other to unlock. For example, consider the following two threads.

- Thread-1

```
acquire(lock1);
acquire(lock2);
[modify stuff]
release(lock1);
release(lock2);
```

- Thread22

```
acquire(lock2);
acquire(lock1);
[modify stuff]
release(lock2);
release(lock1);
```

Here both threads are stuck indefinitely. Thread-1 will be waiting for Thread-2 to release lock1, and Thread-2 will be waiting for Thread-1 to release lock2.

We can prevent deadlock by requiring that locks always be acquired in order. For example, if a thread needs to acquire locks 1, 3, and 2, it must acquire lock1, followed by lock2, followed by lock3.

Answer 1.54: To rotate by $+90$ degrees, we first transpose the matrix then reverse each row. To rotate by -90 degrees, we first transpose then reverse each column of the matrix. The time complexity in both cases is $\mathcal{O}(N^2)$ for an $N \times N$ matrix.

Answer 1.55: We can reverse each row and then reverse each column, or vice versa. The time complexity for an $N \times N$ matrix is $\mathcal{O}(N^2)$.

Answer 1.56: Let's assume we had N number of items in the original dataset, whose mean and standard deviation are μ_{old} and σ_{old}, respectively. Now, we add a new item. The updated values are given by

$$\mu_{\text{new}} = \mu_{\text{old}} + \delta, \tag{A.9}$$

$$\sigma^2_{\text{new}} = \left(\frac{N}{N+1}\right)\left(\sigma^2_{\text{old}} + \delta^2\right) + \left(\frac{1}{N+1}\right)(d-\delta)^2, \tag{A.10}$$

where

$$d = \text{new item} - \mu_{\text{old}}, \text{ and}$$

$$\delta = \frac{d}{N+1} \tag{A.11}$$

Here is a C++ function to compute these.

```
void update(size_t N, const float new_elem, float& mean, float& std) {
    if (N==0) return;
    auto delta = (new_elem - mean)/(N+1);
    auto first_term = N*(std*std + delta*delta)/(N+1);
    auto second_term = pow(new_elem - mean - delta, 2)/(N+1);

    mean += delta;
    stdev = sqrt(first_term + second_term);
}
```

Answer 1.57: REST stands for *Representational State Transfer*. A RESTful API is an application program interface that uses HTTP requests to *GET*, *PUT*, *POST*, and *DELETE* data.

Answer 1.58: REPL stands for Read-Evaluate-Print Loop. It is an interactive interpreter in a programming language such as Python. A program written in a REPL environment is executed piecewise (i.e., one line at a time) during the interactive session.

Answer 1.59: Stateful and stateless are adjectives that describe whether or not an algorithm is designed to remember the past. Stateful algorithms keep track of the state by updating values in a storage field set up for that purpose. Stateless means there is no record of the past and each new record is handled based entirely on information that comes with it. For example, visitor tracking algorithms (e.g., cookies) are stateful while sending/receiving of HTTP packets is stateless.

An advantage of stateful algorithms is that they store data efficiently for future decision making. However, this can also limit their scalability due to their complexity and the CPU or memory limitations. Stateless algorithms can do simple things and generally do not suffer from these limitations.

Answer 1.60: IPv4 addresses are represented as four decimal numbers, each ranging from 0 to 255, separated by dots, e.g., 192.168.1.1. Leading zeros in either of the four parts are not allowed. For example, the address 192.168.01.01 is invalid.

So, to check whether a given IPv4 address is legal, we can parse it as string and check that it has 4 tokens separated by dots. Each token must be a positive integer in the range $0 - 255$ (both inclusive) and should have no leading zeros.

Answer 1.61: It's a philosophical principle, "Everything should be made as simple as possible, but no simpler." So, the best theory is the smallest one that describes all the facts.

Answer 1.62: Moore's law refers to predictions concerning the exponential growth in the density and/or performance of electronic circuits in a processor. A useful summary is that the number of operations per second, per dollar expended, doubles every N months, where N is roughly 18.

Answer 1.63: In-database analytics is a process of performing the analytics directly in the database instead of separate application. This eliminates the overhead of moving large data sets to analytic applications. However, it comes at the cost of reduced flexibility and is often hard to build machine learning on.

Answer 1.64: A a cron job is a scheduled task that runs in the background on a server.

Answer 1.65: Dynamic programming is the process of solving easier-to-solve sub-problems and building up the answer from that. Recursion is the process of a function calling itself, usually with a smaller

dataset. Thus, while dynamic programming is a bottom up approach, recursion is top down. Dynamic programming algorithms could be implemented with recursion, but they don't have to be.

Answer 1.66: A software design pattern is a general reusable solution to a commonly occurring problem. Some best-known design patterns are: listener/observer pattern, singleton pattern, model-view-controller (MVC), etc.

Answer 1.67: We use a singleton pattern when we want to make sure there is exactly one instance of something in the program. For instance, in C++ we can implement singleton class as in the following code:

```
class S {
    public:
        static S& getInstance() {
            static S instance; // Guaranteed to be destroyed
            return instance;
        }
    private:
        S() {} // Constructor

    public:
        S(S const&) = delete; // Don't Implement
        void operator=(S const&) = delete; // Don't Implement
};
```

Answer 1.68: Model-view-controller is a design pattern commonly used in user interfaces. Its goal is to keep the data separate from the user interface. For example, when designing a program to display weather information, the code that downloads the weather data should not depend on the code that displays the information, and vice versa.

Answer 1.69: We need to keep a running count of parentheses. A left parenthesis will increase our count by 1 and a right parenthesis will decrease it by 1. If the count ever goes below 0, it means an unmatched right parenthesis was found. If the string terminates with count > 0, then there is an unmatched left parenthesis.

Here is a C++ function to implement this algorithm.

```
bool balanced(const string& expression) {
    int count = 0;
    for (int i = 0; i < expression.size(); ++i) {
        if (expression[i] == '(') ++count;
        else if (expression[i] == ')') --count;
        if (count < 0) return false;
    }
    return count == 0;
}
```

Answer 1.70: Questions like these are meant to check if you can write a simple programming structure.

Here is a typical implementation in C++.

```
#include <iostream>
#include <string>

Class Person {
public:
  Person(const std::string& name);
  Person(const Person& other);

  std::string getName() const;
  enum class Gender {female, male, unknown};

   struct Address {
     std::string street;
     uint16_t st_num;
     std::string city;
     uint16_t zip;
   };

private:
  std::string m_name;
  uint8_t m_age;
  auto m_gender = Gender::unknown;
  Address m_address;
}
```

Answer 1.71: Floating point precision affects a calculation in profound ways. To begin with, binary floating-point representations are inexact. Simple values like 0.1 cannot be precisely represented using binary floating-point numbers. A slight change in the order of operations or the precision of intermediate storage can change the result.

Answer 1.72: Here is one way to do this:

```
grep -c  ''ABC'' file1
```

Answer 1.73: The command *egrep* is equivalent to "grep -E". The "-E" option switches grep into a special mode so that the expression is evaluated as an ERE (Extended Regular Expression) instead of normal pattern matching. It supports added grep features like "+" (1 or more occurrence of previous character),"?" (0 or 1 occurrence of previous character) and " | " (alternate matching).

Answer 1.74: Here is one way to do this:

```
sed -n '5,$p' myfile | sed '/MNO/s/ABC/DEF/'
```

Answer 1.75: Here is one way to do this:

```
awk '{ total += NF}; END {print total}' myfile
```

Answer 1.76: Here is one way to do this:

```
awk 'NR % 2 == 0' myfile
```

Answer 1.77: Here is one way to do this:

```
awk 'BEGIN {print "Word\tCount";} {word[$0]++;} END {
     for (var in word) print var,"\t",word[var]; }' myfile
```

Answer 1.78: Here is one way to do this:

```
head -99 inputfile | tail -1
```

Answer 1.79: Here is one way to do this:

```
sed -n '10p' inputfile
```

Answer 1.80: Here is one way to do this:

```
sed '/./!d' inputfile
```

Answer 1.81: Here is one way to do this:

```
sed '0~100 s/$/ABCDEF/g' < inputfile > outputfile
```

Answer 1.82: We can use *comm* or *join* command:

```
comm -12 file1 file2
```

Option "-12" will suppress lines that are unique to either file.

```
join file1 file2
```

Answer 1.83: By using the *uptime* command.

Answer 1.84: We can use

```
tail -f filename
```

This will cause only the default last 10 lines to be displayed on screen, thus it will continuously show the updating part of the file.

Answer 1.85: We can use

```
du  -s /home/user1
```

where *user1* is the user for whom the total disk space needs to be found.

Answer 1.86: We can use

```
ps -p ProcessId
```

Answer 1.87: Version control is a tool for managing changes to the source code over time. Some popular version control systems include Git, SVN, CVS, etc., with Git being the most popular. Please describe which ones you have used for things like pull request, merge, code review etc.

Answer 1.88: The interviewer here is looking for your experience with agile processes and practices, e.g., scrum, continuous integration, refactoring, velocity tracking, etc [8].

Answer 1.89: Technical debt refers to cumulative long term costs incurred by cutting corners in a software development. Often it reflects the extra development work that arises when code that is easy to implement in the short run is used instead of applying the best overall solution. Not all technical debt is bad. As with monetary debt, there are often good reasons to take on technical debt.

Technical debt can be mitigated by refactoring code, improving unit tests, reducing dependencies, improving APIs and documentation, etc.

Deploying data driven models in the real world is especially prone to incurring technical debt, because it has all of the maintenance problems of traditional code (which is used for extracting features), plus additional set of issues specific to machine learning. For example, any re-use or chaining of input signals by the model may unintentionally couple otherwise disjoint objects, thus encroaching upon carefully designed software abstraction boundaries[9].

Answer 1.90: The interviewer wants to know if you conscientiously write efficient code and follow test driven development [10].

Writing good code is important because it leads to better performance, low bug count, and easy maintainability [11]. At the heart of it is a code that is easy to read, easy to dissect into components relating to a given change request, and easy to modify without the risk of starting a chain of breakage in dependent modules. Some commonsense principles of writing good code are

- Keep the code as simple as possible.
- Divide and conquer! Complex code can be broken down into multiple modules, each focusing on exactly one thing. This will ensure a clean separation of functionality with high cohesion and low coupling [12].
- Make it readable, especially make sure to document where the main elements are located. Relevant and contextual variable and function names are a great way to do this. Names that convey purpose will even reduce the need for documentation.
- Avoid personal quirks and magic code. The code should be predictable and easy to read in a way people expect.

The interviewer is also checking if you follow good software practices such as using unit and functional tests. One good practice is to write unit-test for each functionality in your code. Run the test and watch it pass, then re-factor the code, and repeat the process [13]. Also, functional tests are a good practice for testing end-to-end functionality.

Answer 1.91: Fermat's Last Theorem states that there are no positive

integers x, y, and z such that

$$x^n + y^n = z^n \tag{A.12}$$

for any integer $n > 2$. However, the cases $n = 1$ and $n = 2$ have infinitely many solutions.

Answer 1.92: We can test whether the generated sequence of random numbers satisfies the following criteria:

- Each bit in a number should have a 50% probability of changing ($1 \to 0$ or $0 \to 1$) when going from one number to the next in the sequence.
- The sequence of numbers should not repeat at all, or if the numbers do cycle, the cycle should be very long.

Answer 1.93: Most typical data structures — such as map, list, hashmap, binary search tree, linked list, etc. — are deterministic. They start with an empty structure and support a sequence of *insert()*, *find()*, and *delete()* operations. The same sequence of operations is guaranteed to always give the exact same resulting structure.

In contrast, in a probabilistic data structure, the same sequence of operations will usually not result in the same structure. The result depends on the sequence of random bits generated during the operations. Generally speaking, these data structures use hash functions to randomize and compactly represent a set of items. Collisions are ignored, but errors can be well-controlled under certain threshold.

These algorithms use much less memory and have constant query time. They can also be easily parallelized for large data sets. Skip lists, HyperLoglog counting, and Bloom filter are some of the most commonly used probabilistic data structures.

Answer 1.94: Bloom Filter is the most famous and widely used probabilistic data structure. It is a bit array of m bits, all initialized to 0. To add an element, we use $k < m$ different hash functions to get k array positions and set the bits at these positions to 1. To query an element, we again use those k hash functions to obtain k array positions. If any

of the bits at these positions is 0, then the element is definitely not in the set. If the bits are all 1, then the element might be in the set.

Bloom filter has the following properties:

- Query time is $\mathcal{O}(k)$.
- False positive is possible when the queried positions are already set to 1. But false negative is impossible.
- Union and intersection of bloom filters with same size and hash functions can be implemented with bitwise OR and AND operations.
- We cannot remove an element from the set.

The optimal number of hash functions, k, can be determined using the formula

$$k = \frac{m}{n} \ln 2, \tag{A.13}$$

where m is the length of the bit array and n is the maximum cardinality (or, capacity). For a false positive rate p,

$$m = -\frac{n \ln p}{(\ln 2)^2} \tag{A.14}$$

Thus, a Bloom filter with 1% false positive rate only requires 9.6 bits per element regardless of the size of the elements.

Answer 1.95: HyperLogLog is a probabilistic data structure for estimating the number of distinct elements in streaming data. It is based on the bit pattern observation that for a stream of randomly distributed numbers, if there is a number x with maximum k leading 0 bits, then the cardinality of the stream is very likely equal to 2^k.

Because a single counter has high variance, in order to get a better estimation, the stream is split into m sub-streams using the first few bits of the hash. Each of the m registers uses at most $\log_2^2 n + \mathcal{O}(1)$ bits when cardinalities $\leq n$ need to be estimated.

The values in the m registers are averaged to obtain the final cardinality estimate. If the standard deviation for each sub-stream is σ, then

the standard deviation for the averaged value is only $\sigma/\sqrt{m}$. With some more optimizations, the algorithm manages to achieve an error rate as low as $1.04/\sqrt{m}$. Thus, HyperLogLog can count one billion distinct items with an accuracy of 2% using only 1.5 KB of memory.

Answer 1.96: Locality-sensitive hashing is an algorithm which hashes input items in such a way that similar items map to the same bucket with high probability. Unlike conventional hashes, a locality-sensitive hashing aims to maximize the probability of collision for similar items. It is a great way to reduce the dimensionality of high-dimensional data.

One example use case where it would be particularly effective is in detecting duplicate items in my address book. Let's assume that my address book consists of a large number of entries manually entered over several years. So, it contains instances of inexact duplicates, which differ in name or ZIP code in minor ways. I want to first detect and then remove these duplicates. I can divide each address into a set of overlapping n-grams, e.g., *MYADDRESS* becomes the set {*MYAD, YADD, ADDR, ..., DRES*}. Then use a hash-table to find colliding n-grams. For example, *MYADDRESS* and *MIADDRESS* will collide because both are associated with n-gram *ADDR*. Now, I can check all such collisions and remove those that I feel are duplicates.

We can improve upon the above procedure by using multiple hash functions and an optimal value of n for the n-grams.

Answer 1.97: A POC is a prototype that is designed to determine the feasibility of a product or methodology, but does not represent deliverables. In engineering, a rough prototype of a new idea is often constructed as a proof of concept.

Answer 1.98: For some reason this question is quite popular. The interviewer probably wants to discuss how an integer is fundamentally stored in memory as 32 bits/4 bytes of information. Or, she meant to ask the maximum integer value that can be stored in a 32-bit system: 2^{32} for *unsigned int*, 2^{31} for *signed int*.

Answer 1.99: This is an open-ended question, because the answer depends on how strictly we want to match the URL. So, you should first define the different parts you want to match in the regex. Each URL has at least three parts

- Protocol: Typically *http*, *https*, or *ftp* followed by an "://"
- Name: Containing certain alpha-numeric characters followed by a dot
- Top Level Domain: For example, *com*, *net*, *io*, etc.

Here is a simple regex to loosely match these three parts

```
(http|https|ftp)?:\/\/(www\.)?[-a-zA-Z0-9@:\%._\+~#=]{2,256}\.[a-z]{2,6}\b([-a-zA
    -Z0-9@:\%_\+.~#?&//=]*)
```

Answer 1.100: This kind of question is meant to stimulate a discussion on how to solve some frequently occurring problem that may have multiple solutions, but each solution imposes potential trade-offs.

To detect an individual account shared by multiple users, you can keep track of logins by location. Multiple logins within a few hours time window from different geographical regions can be a positive indicator. Significantly large number of user sessions or high bandwidth consumption can be other indicators. You can ask for more specifics and identify other such indicators.

Answer 1.101: Some well-known APIs are: AWS API, Google Analytics API, Facebook/Twitter/LinkedIn API to access aggregate users data, GitHub API, etc.

Answer 1.102: Since 90% of the execution time is often spent executing 10% of the code, we can start with optimizing that part of the code. Optimal choice of data structure can certainly help with performance. If the algorithm involves multiple table lookups or access to databases, then an efficient caching can help improve the performance greatly. For some tasks we can improve execution time by doing parallel processing.

Appendix B

Probability and Statistics Answers

This appendix contains answers to the questions posed in Chapter 2. To refresh your knowledge of probability and statistics, please see Ref. [14].

Answer 2.1: 2/3. There are three possible outcomes: boy-girl, girl-boy, girl-girl. Only the first two satisfy the given condition.

Answer 2.2: The number of 6-number combinations from a pool of 49 numbers is ${}^{49}C_6$. The total number of possible combinations from 49 numbers is 49!. Hence, the probability to win a Pick-6 lottery is

$$P_{\text{Pick-6}} = \frac{{}^{49}C_6}{49!} = \frac{6! \times 43!}{49!} = \frac{1}{13983816}, \tag{B.1}$$

i.e., almost one in 14 million.

Answer 2.3: The possible outcomes for two dice totals are listed in Table B.1. Accordingly, the probability of getting sum 4 is 8%, while probability of getting sum 8 is 14%.

Answer 2.4: We can pick one pill from bottle-1, two pills from bottle-2, three pills from bottle-3, and four pills from bottle-4. Their ideal weight should be 10 pills of 10 mg each = 100 mg. Now, let's assume

Total on dice	Pairs of dice	Probability
2	1+1	$1/36 \approx 3\%$
3	1+2, 2+1	$2/36 \approx 6\%$
4	1+3, 2+2, 3+1	$3/36 \approx 8\%$
5	1+4, 2+3, 3+2, 4+1	$4/36 \approx 11\%$
6	1+5, 2+4, 3+3, 4+2, 5+1	$5/36 \approx 14\%$
7	1+6, 2+5, 3+4, 4+3, 5+2, 6+1	$6/36 \approx 17\%$
8	2+6, 3+5, 4+4, 5+3, 6+2	$5/36 \approx 14\%$
9	3+6, 4+5, 5+4, 6+3	$4/36 \approx 11\%$
10	4+6, 5+5, 6+4	$3/36 \approx 8\%$
11	5+6, 6+5	$2/36 \approx 6\%$
12	6+6	$1/36 \approx 3\%$

Table B.1: Possible outcomes for two dice totals.

one of the bottles, say bottle-3, is defective that contains 9 mg pills. Then the total weight of the pills would come out to be 97 mg. So, the difference in the total weight value compared to the non-defect case tells us which individual bottle is defective.

Answer 2.5: Let the expected number of coin flips be x. We have three scenarios:

1. If the first flip is a tail, then we have wasted one flip. The probability of this event is $1/2$ and the total number of flips required is $x + 1$.
2. If the first flip is a head and second flip is a tail, then we have wasted two flips. The probability is $1/4$ and the total number of flips required is $x + 2$.
3. If the first flip is a head and second flip is also head, then we are done. The probability is $1/4$ and the total number of flips required is 2.

Thus,

$$x = (1/2)(x+1) + (1/4)(x+2) + (1/4) \cdot 2, \tag{B.2}$$

which gives $x = 6$.

Answer 2.6: Following the above steps, the answer for 3 consecutive heads is 14 and for n consecutive heads is $2^{(n+1)} - 2$.

Answer 2.7: $N/2$, because the probability for head is $1/2$ for each coin flip.

Answer 2.8: Let the expected number of coin flips be x. Then

$$\begin{aligned} x &= (1-p)(x+1) + p(1-p)\cdot 2 + p^2(1-p)\cdot 3 + ... \\ &= x(1-p) + (1-p)/(1-p)^2 \\ &=_{\text{.}} x(1-p) + 1/(1-p) \end{aligned} \tag{B.3}$$

Thus,

$$x = \frac{1}{p(1-p)}. \tag{B.4}$$

So, for a fair coin the expected number of flips to get a head-tail combination is 4.

Answer 2.9: The number of zeros is equal to the number of times "10" (or "2×5") appears. Since there are a lot more 2s than 5s, it's equal to the number of 5s in the factorization. There is one 5 for every factor of 5 and an extra 5 for 25, 50, 75, and 100. Therefore, we have $20 + 4 = 24$ zeros at the end of 100!.

Answer 2.10: We need to find the minimum n such that

$$(1 - 1/365)^n < 0.5. \tag{B.5}$$

This turns out to be 253 people!

Answer 2.11: Let the expected time be x. Then

$$x = \frac{1}{3} \times 1 + \frac{1}{3}(x+1) + \frac{1}{3}(x+2), \tag{B.6}$$

which gives $x = 4$ hours.

Answer 2.12: We are given

$P(\text{condition}) = 0.001$,
$P(\text{detection} \mid \text{condition}) = 1$,
$P(\text{detection}) = 100\%$ of $0.001 + 5\%$ of $1 = 0.051$.

Following Bayes' theorem,

$$\begin{aligned} P(\text{condition} \mid \text{detection}) &= \frac{P(\text{condition}) \times P(\text{detection} \mid \text{condition})}{P(\text{detection})} \\ &= 0.001/0.051 \\ &= 0.0196, \end{aligned} \tag{B.7}$$

i.e., almost 2%.

Answer 2.13: $\frac{N_M \times \mu_M + N_W \times \mu_W}{N_M + N_W} = \frac{\mu_M + \mu_W}{2}$ if $N_M = N_W$.

Answer 2.14: Given $P(\text{Ferrari}) = 1/3$, $P(\text{red} \mid \text{Ferrari}) = 1/2$.

Following Bayes' theorem,

$$\begin{aligned} P(\text{Ferrari} \mid \text{red}) &= \frac{P(\text{Ferrari}) \times P(\text{red} \mid \text{Ferrari})}{P(\text{red})} \\ &= (1/3) \times (1/2)/1 \\ &= 1/6. \end{aligned} \tag{B.8}$$

Answer 2.15: Following Bayes' theorem,

$$P(\text{raining}|\text{Yes,Yes,Yes}) = \frac{\text{Prior}(\text{raining}) \times P(\text{Yes,Yes,Yes}|\text{raining})}{P(\text{Yes,Yes,Yes})} \tag{B.9}$$

Now

$$\begin{aligned} P(\text{Yes,Yes,Yes}) &= P(\text{raining}) \times P(\text{Yes,Yes,Yes}|\text{raining}) + \\ &\quad P(\text{not-raining}) \times P(\text{Yes,Yes,Yes}|\text{not-raining}) \\ &= 0.25 \times (2/3)^3 + 0.75 \times (1/3)^3 \\ &= 0.25 \times (8/27) + 0.75 \times (1/27). \end{aligned} \tag{B.10}$$

Hence,

$$P(\text{raining} \mid \text{Yes,Yes,Yes}) = \frac{0.25 \times (8/27)}{0.25 \times (8/27) + 0.75 \times (1/27)} = \frac{8}{8+3} = \frac{8}{11}. \tag{B.11}$$

But honestly, you're going to Seattle, so the answer should always be: "YES, I'm bringing an umbrella!"

Answer 2.16: Suppose we drop the first egg from floor n. If it breaks, we will use the second egg from the previous $(n-1)$ floors one-by-one. If it doesn't break, we can drop it from $n + (n-1)^{\text{th}}$ floor. If it survives again, next time we drop from $n + (n-1) + (n-2)^{\text{th}}$ floor, and so on. So, we have the condition

$$n + (n-1) + (n-2) + (n-3) + (n-4) + ... + 1 \geq 100, \quad \text{(B.12)}$$

which gives $n = 14$.

So, our first drop should be from the 14^{th} floor. If the egg survives, we step up 13 floors to floor 27, then up 12 floors to 39, and so on until the first egg breaks. Then go up one floor higher and proceed floor-by-floor dropping the second egg until we find the exact solution.

Answer 2.17: A Bernoulli trial (also called Binomial trial) is a random experiment with exactly two possible outcomes, "success" and "failure", in which the probability of success is the same every time. For example, flipping a coin where the outcome is either heads or tails.The

probability of k successes in n Bernoulli trials is given by

$$P(k) = {}^{n}C_k \, p^k \, (1-p)^{n-k}, \quad \text{(B.13)}$$

where ${}^{n}C_k$ is a Binomial coefficient and p is the probability of success.

When multiple Bernoulli trials are performed, each with a different probability of success, these are referred to as Poisson trials.

Answer 2.18: Probability of k success in n trials = ${}^{n}C_k \, p^k \, (1-p)^{n-k}$, where p is the probability of success in one trial.

Answer 2.19: The probability of observing ≤ 40 heads in 100 coin tosses is given by

$$\sum_{k=0}^{40} {}^{n}C_k \times \left(\frac{1}{2}\right)^k \times \left(\frac{1}{2}\right)^{n-k} = \left(\frac{1}{2}\right)^{100} \times \sum_{k=0}^{40} {}^{100}C_k \approx 0.03. \quad \text{(B.14)}$$

Answer 2.20: The probability of getting 3 left-handed students in a class of 15 with $p = 0.1$ is ${}^{15}C_3 \times (0.1)^3 \times (0.9)^{12} = 0.129$.

Answer 2.21: The Poisson distribution is a discrete probability distribution for a given number of independent events to occur in a fixed interval. For example, if I receive 20 emails per day on average, then the number of emails received on a given day would obey Poisson distribution.

The probability of observing k events in an interval is given by the equation

$$P(k) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{B.15}$$

where λ is the average number of events in the interval.

The Poisson distribution is a limiting case of the Binomial distribution where the number of trials goes to infinity and the expected number of successes remains fixed.

Answer 2.22: We can use Poisson probability formula B.15 here. Because the average event rate is one big earthquake per 100 years, $\lambda = 1$.

$$\begin{aligned} P(0) &= \frac{1^0 \times e^{-1}}{0!} = \frac{1}{e} = 0.368 \\ P(1) &= \frac{1^1 \times e^{-1}}{1!} = \frac{1}{e} = 0.368 \\ P(2) &= \frac{1^2 \times e^{-1}}{2!} = \frac{1}{2e} = 0.184 \\ P(3) &= \frac{1^3 \times e^{-1}}{3!} = \frac{1}{6e} = 0.061 \end{aligned} \tag{B.16}$$

Answer 2.23: We can use Poisson probability formula B.15, where $\lambda =$ average number of chips $= 10$ and $k = 13, 14, \ldots$ etc. Hence the desired probability is

$$\sum_{k=13}^{2000} P(k) = e^{-10} \times \left[\frac{10^{13}}{13!} + \frac{10^{14}}{14!} + \ldots\right] \tag{B.17}$$

Answer 2.24: The president can select three trustees at random out of the total nine trustees in ${}^{9}C_3$ ways. He can have all three chosen trustees agree with him in ${}^{5}C_3$ ways. He can also have just two chosen trustees agree (thus, one disagree) with him in ${}^{5}C_2 \times {}^{(9-5)}C_1$ ways. Hence, the probability that at least two of the chosen trustees will agree with him is

$$\frac{{}^{5}C_3 + {}^{5}C_2 \times {}^{4}C_1}{{}^{9}C_3} = \frac{10 + 10 \times 4}{84} = \frac{50}{84} \approx 59.5\% \tag{B.18}$$

Answer 2.25: We can iterate over the array in order, keeping track of the lowest stock price and the best deal seen so far. Whenever the current stock price minus the current lowest stock price is better than the current best deal, we update the best deal to this new value.

Answer 2.26: We can weigh a set of 3 coins against another set of 3 coins. If one set is heavier, then we weigh one of its coins against another one of its coins. This allows us to identify the heavy coin. If the two sets balance, then we weigh the two leftover coins.

Answer 2.27: For each fair coin the probability of getting 10 heads in a row is $\left(\frac{1}{2}\right)^{10}$. For the faulty coin the probability is 1. Following Bayes' theorem

$$P(\text{Faulty}|\text{Head}) = \frac{P(\text{Head}|\text{Faulty}) \times P(\text{Faulty})}{P(\text{Head})} = \frac{(1/1000)}{P(\text{Head})} \tag{B.19}$$

Now,

$$\begin{aligned} P(\text{Head}) &= P(\text{Head}|\text{Fair}) \times P(\text{Fair}) + P(\text{Head}|\text{Faulty}) \times P(\text{Faulty}) \\ &= \left(\frac{1}{2}\right)^{10} \times \frac{999}{1000} + 1 \times \frac{1}{1000} \end{aligned} \tag{B.20}$$

Thus,

$$P(\text{Faulty}|\text{Head}) = \frac{1}{1 + \frac{999}{1024}} = 0.506. \tag{B.21}$$

Answer 2.28: Let n_1, n_2, n_3 be the number of turns to reach the ant from the starting point, the diagonal corner, and the adjacent corner, respectively. Then

$$\begin{aligned} n_1 &= n_2 + 1 \\ n_2 &= \frac{2}{3}(n_3 + 1) + \frac{1}{3}(n_1 + 1) \\ n_3 &= \frac{1}{3} \times 1 + \frac{2}{3}(n_2 + 1) \end{aligned} \tag{B.22}$$

Solving these we get: $n_1 = 10$, $n_2 = 9$, $n_3 = 7$. So, it will take spider 10 turns to catch the ant.

Answer 2.29: The law of large numbers states that as the sample size grows, its average value gets closer to the average of the whole population. In the case of coin tossing, the law stipulates that the fraction of heads will eventually be close to $\frac{1}{2}$ after a large number of trials.

The law is a building block for much of statistical analysis, i.e., the idea that the mean of a large number of samples can represent the mean of the whole population.

Answer 2.30: The Central Limit Theorem states that the distribution of the mean of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution. The theorem is a fundamental concept in probability theory; indeed it is the reason that many statistical procedures work.

Answer 2.31: The answer to this question would depend on your preferences. For example, I most frequently use scikit-learn.

- Pros
  - Clean and consistent interface
  - Tons of different models implemented
  - Many options for each model but also uses sensible defaults

  - Good documentation
  - Being actively developed
- Cons
  - Relatively new compared to R
  - Initially difficult to learn for someone with non-python background
  - Non-optimal C++ interface to import/export trained model

Answer 2.32: Summary statistics refers to the information that gives a quick and simple description of the data. For example, mean, median, mode, minimum value, maximum value, range, standard deviation, etc.

Answer 2.33: Mean of the sample is $(1+2+3+4+5)/5 = 3$.

Squared deviations of the data points are $4, 1, 0, 1, 4$, respectively. Hence the standard deviation of the sample = $\sqrt{(4+1+0+1+4)/5} = \sqrt{2}$.

Answer 2.34: Since each face is equally likely, each has a probability of $1/6$. Hence, the expected gain = $(1+2+3+4+5+6)/6 = \$3.50$.

Answer 2.35: The normal (or Gaussian) distribution is a probability distribution function (PDF) of a random variable x of the following form

$$N(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\,\sigma}\, e^{-(x-\mu)^2/2\sigma^2}, \tag{B.23}$$

where μ is the mean and σ the standard deviation of the distribution.

Examples: measurement errors, standardized testing scores, yearly rainfall totals, etc.

Answer 2.36: The lognormal distribution is a probability distribution of a random variable whose logarithm is normally distributed. Thus,

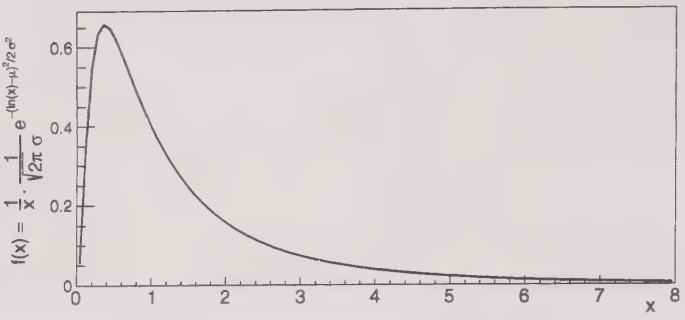


Figure B.1: A lognormal distribution.

if the random variable x is log-normally distributed, then $y = \ln(x)$ has a normal distribution. Likewise, if y has a normal distribution, then $x = e^y$ has a log-normal distribution. Lognormal PDF can be written as

$$N(\ln(x), \mu, \sigma) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-(\ln(x)-\mu)^2/2\sigma^2} \tag{B.24}$$

Figure B.1 shows an example of lognormal distribution.

Answer 2.37: We can perform the following two normality tests

1. Measure goodness of fit of a normal model to the data. If the fit is poor, then the distribution is likely not a normal.
2. Perform hypothesis test on data against the null hypothesis that it is normally distributed.

Answer 2.38: Figure B.2 shows an example of overlapping Normal distributions.

To determine if two Normal distributions of the form B.23: $N(x|\mu_1, \sigma_1)$ and $N(x|\mu_2, \sigma_2)$ overlap, we can compute their product. The region where the product is non-zero would be the overlap region.

Answer 2.39: We first need to find the intersection points between the two distributions $N(x|\mu_1, \sigma_1)$ and $N(x|\mu_2, \sigma_2)$. The overlap region must satisfy the relation

$$\frac{(x-\mu_2)^2}{2\sigma_2^2} - \frac{(x-\mu_1)^2}{2\sigma_1^2} = \ln\left(\frac{\sigma_1}{\sigma_2}\right) \tag{B.25}$$

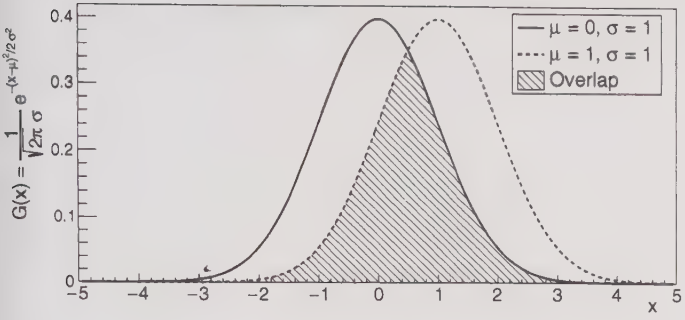


Figure B.2: Overlap between two Normal distributions.

Thus, there are either zero, one, or two intersection points, which divide the overlap region into 1, 2, or 3 parts. We just need to compute area under each part, which is a cumulative distribution function of the corresponding Gaussian. The sum of these areas will give us the overlap between the two Gaussians.

Answer 2.40: Yes. If X and Y are two normally distributed variables, then their sum is also Gaussian.

Let's assume $X = N(x|\mu_1, \sigma_1)$ and $Y = N(x|\mu_2, \sigma_2)$ are independent. Then

$$X + Y = N\left(x|(\mu_1 + \mu_2), \sqrt{\sigma_1^2 + \sigma_2^2}\right). \tag{B.26}$$

So, the sum is Gaussian with its mean being the sum of the two means, and its variance being the sum of the two variances.

If X and Y are correlated, then $X + Y$ is still Normal and the mean is the sum of the means. However, the variances are no longer additive. Instead,

$$\sigma_{X+Y} = \sqrt{\sigma_1^2 + \sigma_2^2 + 2\rho_1\rho_2}, \tag{B.27}$$

where ρ is the correlation. Figure B.3 shows an example of two Normal distributions with their sum superimposed, which is also Normal.

The product of two Gaussians, an example of which is shown in Fig B.4, is generally not a Gaussian.

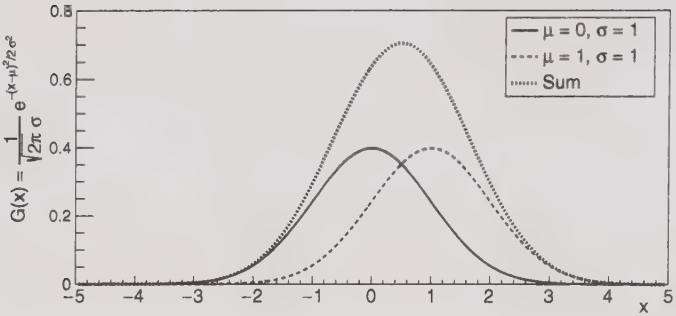


Figure B.3: Sum of two Normal distributions.

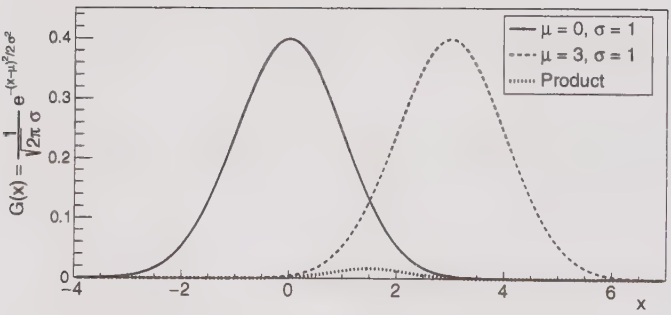


Figure B.4: Product of Normal distributions.

Answer 2.41: Some examples of non-Gaussian distribution are: employee salary distribution in a company, queuing time at a California DMV office, bacteria growth in infested medium, decay of radioactive materials, etc.

Answer 2.42: Student's t-distribution is a probability distributions that arises when estimating the mean of a normally distributed population in situations where the sample size is small and standard deviation is unknown. It has heavier tails than the normal distribution, hence it is more prone to producing values that fall far from its mean. Student's t-distribution has the probability density function given by

$$f(x|\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}, \tag{B.28}$$

where ν is the number of degrees of freedom and Γ is the gamma function. Figure B.5 shows an example of the t-distribution with $\nu = 1$.

Whereas the normal distribution describes a full population, t-distributions describe samples drawn from the full population. Therefore, the t-distribution for each sample size is different, and the larger the sample, the more the distribution resembles the normal distribution.

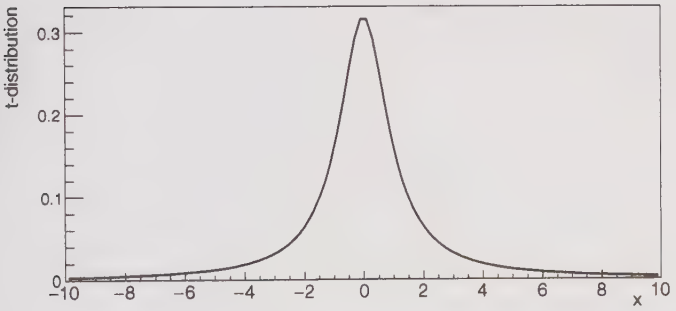


Figure B.5: Example of Student's t-distribution.

Answer 2.43: The F-distribution is a probability distribution that arises frequently as the null distribution of a test statistic, most notably in the analysis of variance. Its probability density function is

given by

$$f(x|d_1,d_2) = \frac{\sqrt{\frac{(d_1\,x)^{d_1}\,d_2^{d_2}}{(d_1\,x+d_2)^{d_1+d_2}}}}{x\,B\left(\frac{d_1}{2},\frac{d_2}{2}\right)}, \tag{B.29}$$

where B is the beta function and the parameters d_1 and d_2 are positive integers. Figure B.7 shows F-distribution for $d_1 = d_2 = 1$.

Answer 2.44: A probability distribution is said to have a long tail, if a larger share of population lies in its tail than would under a normal distribution, as shown in Figure B.6. Examples include many real-world phenomena such as the frequency of a book title sold at Amazon versus time, frequency of internet search terms, citation of journal articles, etc.

They are important because of their usefulness in modeling real-world phenomena.

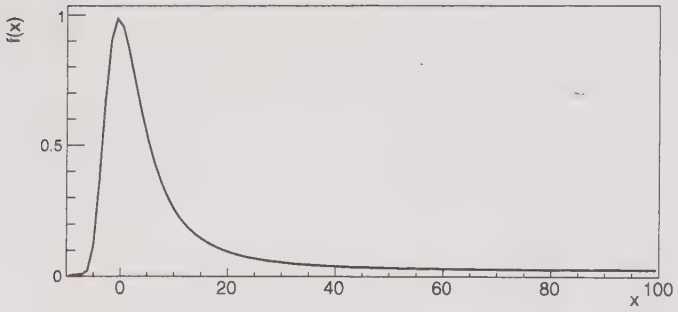


Figure B.6: Distribution with a long tail.

Answer 2.45: The exponential family is a set of probability distributions of a certain form and common mathematical properties. The family includes many of the most common distributions such as normal, exponential, beta, gamma, chi-squared, Poisson, etc.

Answer 2.46: From the rule of three, it can be concluded with 95% confidence that fewer than $\frac{3}{1500} = 0.2\%$ people will experience an adverse event.

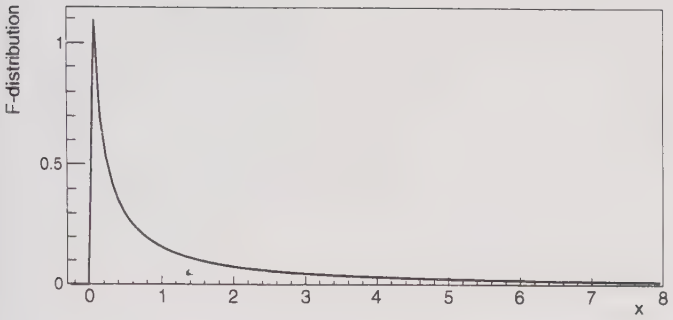


Figure B.7: Example of an F-distribution.

Answer 2.47: A familiar example of chaotic distribution is the leaves falling from trees. Leaves falling from the same place but with a small change in orientation can land in very different places. Intermediate orientations do not necessarily land in between. If the orientation is changed progressively by small amounts, the path of connected landing points becomes quite convoluted, thus showing a strong sensitivity to initial conditions.

Another great example of chaotic system is the weather. In weather forecasting, nonlinear time evolution equations are solved iteratively, each time changing the initial conditions slightly. The distribution of weather outcomes at a given time then provides the envelop of what the forecast will be like.

Answer 2.48: The Dirichlet distribution is the conjugate prior distribution of the multinomial distribution (i.e., the distribution of observed counts of each possible category in a set of categorically distributed observations). The Dirichlet distribution of order $K \geq 2$ with parameters $\alpha_1, ..., \alpha_K > 0$ has the following PDF

$$f(x_1, ..., x_K; \alpha_1, ..., \alpha_K) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{K} x_i^{\alpha_i - 1}, \tag{B.30}$$

where $B(\boldsymbol{\alpha})$ is the multivariate Beta function

$$B(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}, \quad \boldsymbol{\alpha} = (\alpha_1, ..., \alpha_K) \tag{B.31}$$

and $x_1 + ... + x_K = 1$.

Number of trials (N)	95% upper bound for 0 observations
1	0.9500
2	0.7764
3	0.6316
4	0.5271
5	0.4507
10	0.2589
20	0.1391
30	0.0950
50	0.0581
100	0.0295
1000	0.0030
10000	0.0003
100000	0.00003

Table B.2: A 95% upper bound on the rate of occurrence given no observed events in N samples.

Answer 2.49: Here are some popular rules of thumb in statistics.

- The rule of three: If a certain event did not occur in a sample with N subjects ($\hat{p} = 0$), the interval from 0 to $3/N$ is a 95% confidence interval for the rate of occurrences in the population.

  Table B.2 shows 95% upper bounds on the rate of occurrence for such events for some representative values of N, computed directly from the formula

  $$(1 - p)^N \geq 0.05 \tag{B.32}$$

  for the probability of zero observations. When N is greater than 30, $3/N$ is a good approximation to these results.

- One Over Square Root of N rule: A proportion estimate obtained on a sample of N points should only be trusted up to an error of $1/\sqrt{N}$.

- 3-sigma rule: For a normal distribution, almost all data will fall within three standard deviations of the mean — 68% will fall within the first standard deviation, 95% within two standard deviations, and 99.7% within three standard deviations.

Table B.3 lists the z-critical value for some commonly used confidence intervals. The margin of error for a confidence interval is the product of this z-critical value and standard error

$$\text{Margin of Error} = z^* \times \frac{\sigma}{\sqrt{N}}, \tag{B.33}$$

where N is the number of observations.

Confidence level	z^*
50%	0.674
68%	1.0
75%	1.150
90%	1.645
95%	1.960
99%	2.576
99.9%	3.291

Table B.3: The z-critical value for some commonly used confidence intervals.

Answer 2.50: Both describe the degree to which two random variables deviate from their expected values in similar ways. The exact relationship between them is

$$\text{Correlation}(x, y) = \frac{\text{Covariance}(x, y)}{\sigma(x) \times \sigma(y)}, \tag{B.34}$$

where σ denotes the standard deviation.

Thus, the correlation is scaled to be between -1 and $+1$ and is dimensionless. However, the number that represents covariance depends on the units of the data, so it is difficult to compare covariances among data sets that have different scales.

Answer 2.51: By correlation we mean that when one variable increases, the other variable increases (positive correlation), decreases

(negative correlation), or stays the same (no correlation). If one variable is categorical then it doesn't make sense to talk about what happens *when the other variable increases*.

However, if the categorical variable is such that it can be encoded numerically and is monotonically increasing or decreasing, then we can treat it as continuous. Then we can compute the correlation in exact the same way as between two continuous variables.

Answer 2.52: A confounding variable is an extraneous variable that correlates with both the dependent variable and the independent variable. If the researcher fails to control or eliminate confounding variables, they can damage the internal validity of the experiment.

Answer 2.53: The curse of big data is the fact that when we search for patterns in large data sets with billions of data points and thousands of metrics, we are bound to identify coincidences that have no predictive power.

Answer 2.54: The statistical power of a hypothesis test is the probability that the test correctly rejects the null hypothesis (H_0) when the alternative hypothesis (H_1) is true.

Answer 2.55: Root cause analysis is a collective term that describes a wide range of approaches, tools, and techniques used to uncover root causes of faults or problems.

A correlation means relationship between two things — when one increases, the other increases/decreases. A cause is something that results in an effect; for example, heating water to a certain temperature will make it boil. Correlation does not necessarily imply causation. If there is a relationship between two phenomena, A and B, it could be that A causes B, or B caused A, or some other factor caused both A and B, or that they have independent causes that just happen to coincide.

Proving beyond reasonable doubt that A causes B requires more than just a high degree of correlation. Having established a strong correlation, one needs to come up with experiments to test how A might

affect B. If more than one possible cause can be identified, then we conduct experiments in which all but one of the factors remains constant.

Answer 2.56: Recall describes what percentage of true positives are described as positive by the model. Precision describes what percent of positive predictions were correct. The ROC curve shows the relationship between model recall and specificity — specificity being a measure of the percent of true negatives being described as negative by the model.

Recall, precision, and the area under the ROC curve are measures used to identify how useful a given classification model is.

Answer 2.57: Confusion matrix is a table to describe the performance of a classification model for which the true values (i.e., the "ground truth") are known. In the case of a binary classifier, the confusion matrix looks like the following

	Predicted: YES	**Predicted: NO**
Actual: YES	true positives	false negatives
Actual: NO	false positives	true negatives

There are two possible predicted classes: YES and NO. If we were predicting the presence of a disease in a population sample, for example, then YES would mean they have the disease, and NO would mean they don't have the disease. Then there are four possibilities

- **True positives (TP):** These are cases in which we predicted YES (i.e., they have the disease), and they do have the disease.
- **False negatives (FN):** We predicted NO, but they actually do have the disease. (Also known as "Type II error".)
- **False positives (FP):** We predicted YES, but they don't actually have the disease. (Also known as "Type I error".)
- **True negatives (TN):** We predicted NO, and they don't have the disease.

The following quantities can be readily computed using the confusion matrix for a binary classifier

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{B.35}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{B.36}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{B.37}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{FP} + \text{TN}} \tag{B.38}$$

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{B.39}$$

Answer 2.58: Both are bad, but there is an inverse relationship between the number of false positives and the number of false negatives. So you have to trade one off against the other depending on the business cost associated with each.

Answer 2.59: This is the famous "German tank problem" [15]. Let's assume the maximal observed value is m. Then a simple upper bound for the uniform distribution $[0, d]$ for n observations is given by

$$d = m + \frac{m}{n} - 1 \tag{B.40}$$

Answer 2.60: We can go through the elements in order, swapping each element with a random element in the array that does not appear earlier than the element. This takes $O(n)$ time.

Answer 2.61: We need a hypothesis test to evaluate two mutually exclusive statements about a population to determine which statement is best supported by the data. When we say a finding is statistically significant, it's thanks to a hypothesis test.

Answer 2.62: The null hypothesis refers to a general statement or default position that there is no relationship between two measured

phenomena. Often it is compared with an alternative hypothesis in a statistical hypothesis test.

Answer 2.63: The *p*-value [16] is the probability of observing the current value (or a more extreme value) for the test statistic given that the null hypothesis is true. In a hypothesis testing, researcher establishes the criterion for accepting or rejecting the null hypothesis before collecting data. If she chooses the 1% level, it means that if her test statistic is in the rejection region, there is only a 1% chance she would obtain this test statistic if the null hypothesis is true.

In very large samples, *p*-values can quickly go to zero, and solely relying on *p*-values can lead the researcher to claim support for results of no practical significance. For this reason it is important to quote the sample size along with the *p*-value when reporting result.

Answer 2.64: *p*-value is uniformly distributed between 0 and 1 when the null hypothesis is true. By contrast, if the alternative hypothesis is true, the distribution is dependent on sample size and the true value of the parameter being studied.

Answer 2.65: A t-test is a hypothesis test in which the test statistic follows the t-distribution under the null hypothesis. It is most commonly applied when the test statistic would follow a normal distribution, but is unknown and estimated based on a limited data sample.

An F-test is also a hypothesis test in which the test statistic follows the F-distribution under the null hypothesis. It is most often used when comparing statistical models that have been fitted to a data set, in order to identify the model that best fits the population from which the data were sampled. For example, to check the hypothesis that the means of a given set of normally distributed populations, all having the same standard deviation, are equal.

Analysis of variance (ANOVA) is a procedure to analyze the differences among group means and their variations. The observed variance in a particular variable is partitioned into components attributable to different sources of variation. In its simplest form, ANOVA provides

a statistical test of whether or not the means of several groups are equal, and therefore generalizes the t-test to more than two groups.

Answer 2.66: We can perform t-test to assess whether the means of the two groups are statistically different from each other by computing

$$t = \frac{|\mu_1 - \mu_2|}{\sqrt{\sigma_1^2 + \sigma_2^2}}. \tag{B.41}$$

If $t > 0.5$, then the two means are statistically different. If we have 3 or 4 populations we can do pairwise t-test, but ANOVA would provide a more robust result.

In the ANOVA setup, we compute the normalized difference as

$$t_i = \frac{|\mu_i - \mu_{\text{grand}}|}{\sigma_{\text{grand}}} \tag{B.42}$$

for each group i, where

$$\mu_{\text{grand}} = \frac{\sum_{i=0}^{k} \mu_i}{k}, \quad \sigma^2_{\text{grand}} = \frac{\sum_{i=0}^{k} \sigma_i^2}{k-1} \tag{B.43}$$

are the grand mean of all group means and the combined variance, respectively, and k is the number of groups. If $\sum_i t_i/k > 0.5$, then we can assume the two means are statistically different.

Answer 2.67: Following the above procedure, we can compute t_i for each population using Eq. B.42. We then examine each t_i individually and identify which ones are greater than our tolerance value of 0.5.

Answer 2.68: A confidence interval is a range of values which act as good estimates of the unknown parameter in the observed data. For example, if we construct 95% confidence intervals in an infinite number of independent experiments, then the proportion of those intervals that contain the true value of the parameter will be 95%.

Please note that the interval computed from a particular sample does not necessarily include the true value of the parameter, because the

observed data are random samples from the true population. Similarly, a 95% confidence interval does not mean that 95% of the sample data lie within the interval. It merely means that there is a 95% probability that the interval covers the true value of the parameter.

Confidence intervals are very useful in significance testing. For example, let's say we want to test the null hypothesis $\theta = 0$ against the alternative that $\theta \neq 0$. We can do this by determining whether the confidence interval for θ contains 0 for some pre-determined confidence level (e.g., 95%).

Answer 2.69: A point estimate is a single value estimate of the population parameter, derived using a random sample of the population. The confidence level describes the uncertainty of the sampling method. For example, the sample mean $\langle x \rangle$ is a point estimate of the population mean μ. We can compute a 95% confidence interval for μ. This means that if we used the same sampling method to select different samples, the true μ would fall within a range defined by the sample mean $\pm$ the margin of error, 95% of the time.

Answer 2.70: Let's assume that the non-parametric distribution is given as a histogram, with true mean μ. We can generate a large number of samples by bootstrapping from this distribution. We then take the bias-corrected mean of these samples, $\langle x \rangle$, as the test statistic and estimate the margin of error e such that

$$\text{sample mean} - e < \mu < \text{sample mean} + e \tag{B.44}$$

occurs 95% of the time (or any chosen confidence value). This gives an asymmetrical confidence interval around $\langle x \rangle$.

Answer 2.71: Randomization is important because it minimizes any possible biases that may arise in the experiment and helps control the effect of lurking variables.

Answer 2.72: Power analysis can be used to calculate the minimum sample size required so that one can be reasonably likely to detect an effect of a given size. The following four quantities are closely related:

1. sample size
2. effect size
3. significance level = P(Type I error) = probability of finding an effect that is not there
4. power = $1 - P$(Type II error) = probability of finding an effect that is there.

Given any three, we can determine the fourth.

Answer 2.73: We could estimate the probability of an event-type that occurred r times in past N trials as

$$P(r) = \frac{r}{N} \tag{B.45}$$

This works well if r is large. But as r gets smaller, the estimate gets worse. If $r \to 0$, it may still be quite unwise to bet that the event-type in question will never occur in future. More importantly, the fraction of future events whose past counts are zero may be substantial. So the trick is to somehow enhance $P(r)$ from r/N to a higher value in one of the following ways

- using Monte Carlo simulation of unseen rare events, or
- assigning higher weight to rare events seen in the past.

Answer 2.74: Extrapolation is an approximation based on extending a known sequence of values beyond the range that is known with certainty. Interpolation is an estimation of a value within two known values in a sequence.

Answer 2.75: We need to sample because we usually cannot gather data from the entire population. Even in relatively small populations, the data may be needed urgently, and including everyone in the population in our data collection may take too long. We can do random sampling. The required sample size can be determined based on the precision requirement for the experiment and the choice of confidence interval.

Answer 2.76: Some possible forms of bias are

- Selection bias
- Under-coverage bias
- Survivorship bias, i.e., logical error in deriving conclusions from the surviving events since no other events survived.

Answer 2.77: Selection bias is the sampling of data for analysis in such a way that proper randomization is not achieved, thereby ensuring that the sample obtained is not representative of the population intended to be analyzed.

We can try to avoid obvious sources of bias during the sampling process, e.g., time ordering, self-selection etc. But in general, selection bias cannot be overcome with statistical analysis of existing data alone. We need to examine correlation between some external indicator and an internal variable.

Answer 2.78: When performing Bayesian inference, we need the posterior probability distribution over a set of random variables[17]. Unfortunately, this often requires calculating intractable integrals. In such cases, we may give up on solving the analytical equations and proceed with Markov Chain Monte Carlo (MCMC) sampling instead. Some of the popular sampling techniques are:

- Accept-reject method
- Gibbs sampling
- Metropolis-Hastings algorithm
- Importance sampling.

For more detail on MCMC methods, see Ref [18] and Ref [19].

Answer 2.79: In the accept-reject method, we first generate a sample from some known distribution $f(x)$ having upper bound M and then decide whether to accept or reject based on whether the generated value is less than the target posterior $p(x)$. The method is very simple and intuitive. However, the main problem with this process is that M is generally large in high-dimensional spaces, so

$$p(\text{accept}) \propto \frac{1}{M} \implies \text{very small} \tag{B.46}$$

Thus, most samples will likely get rejected. Hence, this methods can be inefficient in computing time.

Answer 2.80: In Gibbs sampling, we first analytically derive the posterior conditionals, $p(x_i|x_1,..,x_{i-1},x_{i+1},..,x_n)$, for each random variable x_i

$$p(x_i|x_1,..,x_{i-1},x_{i+1},..,x_n) = \frac{p(x_1,..,x_n)}{p(x_1,..,x_{i-1},x_{i+1},..,x_n)} \tag{B.47}$$

For example, in the case of three random variables x_1, x_2, x_3, we compute $p(x_1 \mid x_2, x_3)$, $p(x_2 \mid x_1, x_3)$, and $p(x_3 \mid x_1, x_2)$ using Eq. B.47. Then we simulate posterior samples from the target joint posterior by iteratively sampling a value for a random variable from its corresponding posterior conditional, while all other variables are fixed to their current values. Unlike in the accept-reject method, all proposed samples are accepted.

The technique often suffers from slow mixing and could take a long time to sufficiently explore the values of x in order to give a good characterization of the posterior.

Answer 2.81: Suppose we want to sample from a distribution $P(x)$. In Metropolis-Hastings (MH) algorithm, we iteratively generate a sequence of sample values from a proposal distribution $f(x) \propto P(x)$ in such a way that, as the sample size $\to \infty$, the distribution of values closely approximates $P(x)$. At each iteration, the algorithm picks a candidate for the next sample value, x', based on the current sample value x_t. The candidate is accepted with probability

$$\alpha = \frac{P(x')}{P(x_t)} = \frac{f(x')}{f(x_t)} \tag{B.48}$$

If the candidate gets accepted, its value is used in the next iteration, otherwise the current value is reused in the next iteration. Popular choices of proposal distribution include Gaussian, uniform, and log-normal. Since the proposal distribution randomly perturbs the current state of the chain, and then either accepts or rejects the perturbed value, the algorithms is also called "random-walk Metropolis algorithm".

With MH method we can sample from any distribution $P(x)$, provided we can compute the value of a function $f(x)$ that is proportional to $P(x)$. The lax requirement that $f(x)$ should be merely proportional to the true density, rather than exactly equal to it, makes MH particularly useful in situations where the normalization factor is not known.

Answer 2.82: In importance sampling method, we sample from a wrong but easy to generate distribution $g(x)$. Then we compensate by an importance weight

$$w(x) = \frac{f(x)}{g(x)}, \tag{B.49}$$

where $f(x)$ is the desired correct distribution.

Importance sampling is often used to estimate posterior parameter expectations in problems that are too hard to treat analytically, for example in Bayesian networks. The method is also a handy tool for computing integrals of type $\int f(x)\, p(x)\, dx$, which then evaluates to

$$\int f(x)\, p(x)\, dx = \frac{\sum_{i=0}^{N} p(x^i) w(x^i)}{\sum_{i=0}^{N} w(x^i)} \tag{B.50}$$

Answer 2.83: Mathematically speaking, Gibbs sampling is a special case of MH sampling, where proposal distributions are the posterior conditionals. Since all proposals are accepted in Gibbs sampling, the acceptance probability is always 1.

Answer 2.84: Importance sampling as a variance reduction technique is widely used in Monte Carlo simulations. The idea here is that certain values of the input variables in simulation have more impact on the parameter being estimated than others. If these values are sampled more frequently, by choosing appropriate $g(x)$ in Eq. B.49, then the estimator variance will be reduced.

This use of "biased" distributions can result in a biased estimator if applied directly. However, the simulation outputs are weighted to

correct for this effect. The weight is given by the likelihood ratio of the true underlying distribution with respect to the biased simulation distribution.

Answer 2.85: We should use MCMC methods in this case. The reason is that most accept-reject sampling methods suffer from the "curse of dimensionality", where the probability of rejection increases exponentially with the number of dimensions. MCMC methods do not suffer from this problem to such a degree. As a result, MCMC methods are the preferred choice when the number of dimensions is high.

Answer 2.86: Although MCMC sampling methods eventually converge to the desired distribution, the initial samples may follow a very different distribution, and therefore may not accurately represent the desired distribution. This is especially the case when the starting point is in a region of low density. As a result, a burn-in period is typically necessary, where an initial number of samples (e.g., the first 1000 or so) are thrown away.

Answer 2.87: Resampling is a method for estimating sample statistics — means, medians, variances, etc. — using one of the following techniques:

- **Jackknifing:** Using subsets of available data.
- **Bootstrapping:** Drawing randomly with replacement from data.

One difference between the two is that bootstrapping gives different results when repeated on the same data, whereas jackknifing gives exactly the same result each time. Thus, jackknifing is good for quickly estimating the variance of a point estimator. Bootstrapping, on the other hand, is good for estimating the whole sample distribution when needed, then computing variance from that. It is typically computer intensive.

Answer 2.88: Suppose we want to perform a hypothesis test between the null hypothesis H_0 and an alternative hypothesis H_1, where the confidence interval α (e.g., 0.05) and test statistic (e.g., sample mean) are known. We now need the p-value, but to compute this we need to know the sampling distribution of our test statistic when the null hypothesis is true.

Permutation test gives us a simple way to compute the sampling distribution for the test statistic under the null hypothesis. We can simulate a large data sample under the null hypothesis conditions, then resample from this simulated data. The ranking of the real test statistic evaluated based on this distribution gives us the p-value.

Answer 2.89: Since $n > 30$, the distribution of sample means can be approximated by a normal distribution. Let's assume

- The null hypothesis H_0: Mean body temperature of healthy adults $\mu = 98.6°F$.
- Alternative hypothesis H_1: $\mu \neq 98.6°F$.

The test is two-sided because a sample mean significantly less than or greater than 98.6°F would be strong evidence against the null hypothesis.

The z-value is

$$z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}} = \frac{98.20 - 98.6}{0.62/\sqrt{106}} = -6.64 \tag{B.51}$$

For $\alpha = 0.05$, we would fail to reject H_0 if $-1.96 < z < 1.96$. But since $z = -6.64$, we reject H_0, i.e., the hypothesis that the mean body temperature of healthy adults is equal to 98.6°F.

Answer 2.90: A/B testing is a statistical hypothesis testing for randomized experiment with two variables A and B. The objective is to detect any changes in these variables to maximize the outcome of an interest. This is different from comparing an alternative hypothesis with the null hypothesis.

Answer 2.91: Here the interviewer is interested in the details of your improvement, what kind of improvement metric was used, how the A/B testing was designed, etc.

Answer 2.92: Here the interviewer is looking for intuition regarding the choice of appropriate metric and also what would factor in when designing A/B tests in each case.

Answer 2.93: A common pitfall in performing A/B testing is the habit of looking at the test result while it's running, then stopping the test as soon as the p-value reaches a particular threshold, say, 0.05. This renders the p-value fundamentally untrustworthy, because it often shows spuriously high significance for new features that offer no improvement.

One solution is to pre-commit to running the A/B testing experiment for a particular amount of time, never stopping early or extending it farther. Another solution is to rely on a Bayesian procedure rather than frequentist hypothesis testing. Bayesian methods are less susceptible to this problem, because during the test's running we can update our model (i.e., determine a new posterior distribution) based on the data observed so far[17]. At any point in time we can use the updated model to determine if our observations support a winning conclusion, or if there still is not enough evidence to make a call.

Answer 2.94: The standard deviation of a sequence of values is zero if and only if all the values are equal to the mean. So

$$x_1 = x_2 = \cdots = x_n = \mu = \frac{x_1 + x_2 + \cdots + x_n}{n} \tag{B.52}$$

Answer 2.95: Time series is a sequence of discrete-time data. Examples include heights of ocean tides, monthly rainfall numbers, daily sunrise times etc. Time series data generally have two basic patterns — trend and seasonality. The former represents a systematic linear or (most often) nonlinear variation over time. The latter repeats itself in systematic intervals over time.

Answer 2.96: Unlike the usual statistical analyses, the analysis of time series is based on the assumption that successive values in the data represent consecutive measurements taken at equally spaced time intervals.

Answer 2.97: Here you need to describe what kind of transformations you needed to make, was it a time series modeling and prediction

problem, did you need to sample from the time series distribution, how did you deal with delayed arrival data, etc.

Answer 2.98: Resampling involves changing the frequency of the time series data. In the case of upsampling (e.g., from minutes to seconds), we need to determine how fine-grained the resampled observations need to be and how to calculate them using interpolation. In the case of downsampling, we need to determine the appropriate summary statistics (e.g., mean, median, or some kind of spline) to calculate the new aggregated values.

Answer 2.99: A sliding window is a sub-list that runs over an underlying collection. For example, with collection [a b c d e f], a sliding window of size 3 would run over it like [a b c], [b c d], [c d e], [d e f]. Internet's TCP protocol uses sliding time windows to transmit data packets.

Answer 2.100: We can compute the z-value for each condition and then use it to compute the cumulative probability.

1. If one man is randomly selected:
   $z = (x - \mu)/\sigma = (180 - 173)/30 = 0.23$.
   For normal distribution, $P(z > 0.23) = 0.4090$.
2. If 36 different men are randomly selected:
   $\sigma_{\bar{x}} = \sigma/\sqrt{N} = 30/\sqrt{36} = 5$. Therefore, $z = (180 - 173)/5 = 1.40$.
   For normal distribution, $P(z > 1.4) = 0.0808$

Answer 2.101: Multiple seasonality can be modeled using Fourier series with different periods. If there are two seasonal variations, weekly and yearly, with respective periods of 7 days and 365 days, then we need to include corresponding Four series terms for each.

Answer 2.102: An algorithm should be updated based on how quickly the underlying data distribution evolves. So, it is an optimization between the cost of updating the algorithms vs. the degradation in the performance of the current version. However, if the new version is

qualitatively better (e.g., more features, better performance etc.) then it should be deployed sooner.

Answer 2.103: Bayesian and Frequentist approaches differ fundamentally in how they interpret probability. The frequentist defines an event's probability as the limit of its relative frequency in a large number of trials.

$$P_{\text{Frequentist}}(\text{Event}) = \lim_{n\to\infty} \frac{k}{n}, \tag{B.53}$$

where k is the number of successes and n is the number of trials. In this approach, it doesn't make sense to associate probability distribution with a parameter.

The Bayesian defines an event's probability using Bayes' theorem, based on prior knowledge of conditions that might be related to the event:

$$P_{\text{Bayesian}}(\text{Event}|\text{Data}) = \frac{P(\text{Event}) \times P(\text{Data}|\text{Event})}{P(\text{Data})}, \tag{B.54}$$

where $P(\text{Event})$ is a prior belief about the event's occurrence, $P(\text{Data}|\text{Event})$ is the conditional probability of getting data given that the event occurred, and $P(\text{Data})$ is the probability of getting data. In the case of k successes in n trials, the ratio $P(\text{Data}|\text{Event})/P(\text{Data})$ is a function of k and n. Therefore

$$P_{\text{Bayesian}}(\text{Event}|\text{Data}) = P(\text{Event}) \times f(k,n) \tag{B.55}$$

For uniform prior and a large number of trials, $P(\text{Event}) = 1$ and $f(k,n) \approx k/n$, so Eq. B.53 and Eq. B.55 converge to the same value. But, in general, this is not guaranteed.

Answer 2.104: Data modeling is another area where Bayesians and frequentists have fundamental differences. The Bayesian attempts to model probabilistically the parameters that govern the distribution in data, while the frequentist attempts to model the data itself. As a result, Bayesian approach often leads to an answer that is a probability statement, e.g., *there is a* 95% *probability that the success rate in a coin-toss is between* 0.4 *and* 0.6. The frequentist approach to the same problem

gives an answer in the form of confidence level, e.g., *with* 95% *confidence level in the observed data, the success rate in a coin-toss is between* 0.4 *and* 0.6.

Answer 2.105: Personally, I believe the answer to this question depends on the task at hand. I will certainly use Bayesian techniques if there is a reasonable prior, and a close form solution for the posterior exists that is computationally fast. On the other hand, if the task involves comparing two random data sets, I would use frequentist techniques, such as χ^2-test or F-test. Then there are tasks, such as hypothesis testing, for which I often end up using a combination of the two approaches. A common use case is to construct frequentist confidence intervals for some unknown parameter in the Bayesian model.

Answer 2.106: The covariance matrix is not positive definite because it is singular. This means that at least one of my input variables can be expressed as a linear combination of the others. So, I need to remove such variable(s) from the PDF.

I can start by first removing all variables then adding them sequentially and checking the covariance matrix at each step. If a new variable creates singularity, then drop it and go on to the next one. Eventually I should have a subset of variables with a positive definite covariance matrix.

Appendix C

Machine Learning Answers

This appendix contains answers to the questions posed in Chapter 3.

Answer 3.1: Machine Learning is a fairly generic technique for deriving intelligence from data[20]. It can be applied to any kind of data.

Data mining, on the other hand, is a process to extract interesting patterns from *unstructured data*. Machine learning can be used in the data mining process[21].

Answer 3.2: Some popular ubiquitous applications of ML are

- Amazon, Netflix, YouTube, ... recommender systems
- Google translation
- Facebook news feed & friend recommendations
- Spotify, Pandora etc.

You can pick one of these and describe in some detail how it works.

Answer 3.3: Here the interview is trying to test your communication skills. Just pick an ML algorithm you are most familiar with. Then describe it in simple terms using diagrams and chart as necessary.

Answer 3.4: Here the interviewer is looking for both intuition and specific evaluation techniques. Make sure to emphasize your experience with feature engineering from past projects.

Answer 3.5: In supervised learning the data categories are labeled. Unsupervised learning assumes unlabeled data. Table C.1 lists some key differences between the two.

	Supervised	Unsupervised
Target classes	Known	Unknown
Data needs	Labeled samples	Unlabeled samples
Based on	Training set	No prior knowledge
Use case	Predict future data	Explore current data
Primary tasks	Classification, regression	Clustering, PCA, ...
Algorithms	Naïve Bayes, SVM, neural nets, etc.	*K*-means, DBSCAN, optimizations, etc.

Table C.1: Supervised vs. unsupervised learning.

Answer 3.6: When I do statistics, I want to infer how the data was generated. When I do machine learning, I want to predict what the future data will look like. Knowing how the data was generated can still give useful hints about what a good predictor would be. So there is a good deal of overlap between statistics and machine learning.

Answer 3.7: Deep Learning [22, 23] is an offshoot of ML with its origin in Artificial Neural Networks [24]. These days we use neural networks with more complex structures, which are getting deeper, but fundamentally they are still the same old beast. So, I think Deep Learning fits firmly in the ML field.

Answer 3.8: The statement is a gross generalization. Only one part of ML, namely regression, has similarities with curve fitting. ML is more than just regression. Classification and clustering don't have much in common with curve fitting.

Answer 3.9: "Inductive machine learning" is same as supervised learning.

Answer 3.10: You can name any five, e.g., Naïve Bayes, neural networks, decision trees, random forest [25], *K*-means clustering, etc. See Table C.2 for a basic comparison of the supervised ML algorithms.

Algorithm	Problem Type	Accuracy	Training Speed
Linear regression	Regression	Low	Fast
Logistic regression	Classification	Low	Fast
Naïve Bayes	Classification	Low	Fast
Fisher LDA	Classification	Low	Fast
Decision trees	Either	Low	Fast
SVM, kernal tricks	Either	High	Slow
Random forest	Either	High	Slow
AdaBoost	Either	High	Slow
Neural nets (MLP)	Either	High	Slow

Table C.2: Supervised Machine Learning algorithms.

Answer 3.11: Genetic programming is a method that genetically breeds a population of computer programs to solve a problem. It starts with a population of randomly generated computer programs, then iteratively breeds and applies natural selection to successive generations based on predefined fitness criteria, and finally stops when the termination criterion is satisfied. The single best program in the population produced during the run is designated as the solution.

Answer 3.12: Recommender system is an information filtering systems to predict user preferences, given the preferences of other similar users and products.

Answer 3.13: Collaborative filtering is a technique used by recommender systems. In collaborative filtering, algorithms are used to make automatic prediction about a user's interests by compiling preferences from several users.

Answer 3.14: Cosine similarity is a measure of similarity between two vectors $\vec{v}_1$ and $\vec{v}_2$ based on the cosine of the angle between them:

$$\text{similarity} = \cos(\theta) = \frac{\vec{v}_1 \cdot \vec{v}_2}{|\vec{v}_1|\,|\vec{v}_2|} \tag{C.1}$$

Thus, two vectors with the same orientation have a cosine similarity of 1 and two vectors at 90° have a similarity of 0, independent of their magnitude.

Answer 3.15: Some popular libraries available in Python are *numpy*, *scipy*, *pandas*, *scikit-learn*, *matplotlib*, *TensorFlow*, etc.

Answer 3.16: You can briefly describe your use cases with some context.

Answer 3.17: This question gives you a chance to dive deep into some of your favorite features available in these libraries.

Answer 3.18: *TensorFlow* is a low-level library with support for ML building blocks such as vectorization, matrix operations, gradient descent, optimization for GPU, etc. On the other hand, *scikit-learn* is a high-level library that comes with several off-the-shelf algorithms, so one can simply define a model in a few lines of code, then fit it to data, or use for prediction.

Answer 3.19: A lot of research has gone into making some frequently used numerical operations, such as matrix addition and multiplication, fast. The idea of vectorization is that we would like to express our learning algorithms in terms of these highly optimized operations, instead of nested loops. Vectorization is not only simpler, but also runs much faster. It often accomplishes the same task by optimizing computational complexity and eliminating intermediate steps.

Answer 3.20: *Torch*, *TensorFlow*, and *Theano*[26] are low-level library with support for ML building blocks such as vectorization, matrix operations, gradient descent, optimization for GPU, etc. On the other hand, *Caffe* and *Keras* are higher level libraries with implementation of popular Deep Learning frameworks. By asking this question the interviewer is interested to know your familiarity with these libraries, their pros & cons, quirks, latest developments, etc.

Answer 3.21: These are C++ based libraries for numerically solving linear algebra problems and function minimization. If you have a

prior experience dealing with them, then this is a good time to showcase your knowledge.

Answer 3.22: There are several scenarios, a few of them being

1. To adapt an algorithm for the specific needs of our application.
2. To implement a recently discovered library that hasn't yet made into the ML library.
3. To optimize for speed, e.g., in parallel computing architecture.

Answer 3.23: Using a topic modeling algorithm, such as the Latent Semantic Indexing. An efficient LSI graph can be made based on recent historical data.

Answer 3.24: This can be a long-winding discussion, not unlike system design questions. So, try to come up with a few features, such as

- People you currently follow
- Your location
- The accounts followed by the people you follow
- How other users express interest in Tweets from these accounts, etc.

Then explain in some detail how exactly you would use these features.

Answer 3.25: Here the interviewer is interested in your familiarity with ML tools available on popular cloud platforms, such as AWS, Azure, or Google cloud. Although they are similar to their stand-alone version, often the allowed usage is more restrictive and sometimes have well-known quirks.

Answer 3.26: Gradient descent, also known as "steepest descent", is an algorithm that minimizes functions. Given a function defined by a set of parameters, gradient descent starts with an initial set of parameter values and iteratively moves toward a set of parameter values that minimizes the function.

Answer 3.27: No, gradient descent methods do not always converge to the same point. This is because in some cases the algorithm reaches a local minima. In such cases the convergence to the local minima is driven by the starting conditions.

Answer 3.28: A convex function has exactly one minimum and the line segment between any two points on the graph of the function lies above or on the graph. Hence, the convex functions are very useful in optimization problems[27]. A non-concave function is either linear or convex, which doesn't have these nice properties[28].

Answer 3.29: Two commonly used calibration methods are: Platt scaling and isotonic regression. To prevent overfitting on a small calibration set, Platt scaling is better because parameters are more constrained. For a large calibration sample, isotonic regression typically performs better.

Answer 3.30: If the loss function is strictly convex, it is guaranteed to have a unique global minimum. But a non-convex loss functions may have several local minima, i.e., points where the loss function has the minimum value in the local neighborhood, but that is not the global minimum.

A local minimum is not necessarily bad if its loss value is very similar to the global minimum.

Answer 3.31: In gradient descent you need to run through ALL entries in the training data to do single parameter update in a particular iteration[29]. In stochastic gradient descent, however, you can use ONE training entry to update a parameter in a particular iteration. Stochastic gradient descent often converges much faster compared to gradient descent, but the error function may not be as well minimized (due to, e.g., local minima).

Answer 3.32: Yes, regression is a supervised learning, because we need labeled training data set for regression.

Answer 3.33: Some possible problems with regression models are

- Bias
- High variance
- Too many unknown parameters and not enough data points to constrain them
- Highly correlated features, etc.

They can be ameliorated by better feature selection, normalizing the features, and using a regularization procedure.

Answer 3.34: Ordinary least squares is a method for estimating unknown parameters in a linear regression. The method learns by minimizing the sum of squared differences between the observed responses and prediction. If the errors are uncorrelated and normally distributed, then OLS is the maximum likelihood estimator.

Answer 3.35: Let's assume our data set is a collection of N points (x_i, y_i). Our regression model is $h(x, \theta)$ with parameter coefficients $\theta = (\theta_0, \theta_1, \theta_2, ...)$. Then, we want to minimize the cost function

$$J(\theta) = \frac{1}{2N} \sum_i \left(h(x_i, \theta) - y_i\right)^2 \tag{C.2}$$

We can solved this analytically by taking partial derivative of $J(\theta)$ with respect to each parameter and equating to zero

$$\frac{\partial J(\theta)}{\partial(\theta_j)} = 0, \tag{C.3}$$

for each parameter θ_j[29]. In real life we solve it numerically using gradient descent.

Answer 3.36: It is a regression that is linear in the unknown parameters used in the fit, i.e.,

$$y = \sum_i w_i x_i + b, \tag{C.4}$$

where the parameters w_i are linear. Linear regression is called linear because the dependent variable is modeled as a linear combination of inputs and weights.

Answer 3.37: For some reason this question is asked very frequently in interviews.

In contrast to the linear regression, where the output is continuous, the output in the logistic regression is binary.

Answer 3.38: Linear and logistic regressions are the two simplest types of regression. See the answer to question 3.37 for difference between them.

Answer 3.39: We can compute a feature's importance as the ratio of its coefficient to the estimated error in that coefficient. We can then rank each feature by their importance.

Answer 3.40: There are four major assumptions:

1. There is a linear relationship between the variables.
2. The errors or residuals of the data are normally distributed and independent from each other.
3. There is minimal multicollinearity between explanatory variables.
4. Variance around the regression line is same for all values of the predictor variable.

Answer 3.41: We need to keep in mind the major assumptions of linear regression when using it, since they impose significant constraints. These are listed in the answer to question 3.40. In addition, linear regression would do an inadequate job of modeling the data if the data is nonlinear.

Answer 3.42: If some of the linear regression assumptions are violated, then either the bias or the variance of the estimate (or both) will increase. But in practice, the model may still work well in a wide range of the input values.

Answer 3.43: Yes, it is a linear model. Because we can write it as

$$Y \sim (X_1 + X_2)^2 - X_1 X_2, \tag{C.5}$$

which is linear in the transformed variables $(X_1 + X_2)$ and $X_1 X_2$.

Answer 3.44: The variance of the error term generally decreases with inclusion of additional regressors.

Answer 3.45: If the algorithm trains a large number of independent learners with low bias and high variance (i.e., weak learners) and combines their result, then the combined learner is likely to have low bias and low variance, hence a very desirable outcome.

Answer 3.46: If we know from the context of the problem that output is always zero when all inputs are zero, then we do not need intercept term. Otherwise we do.

Answer 3.47: Ordinary Least Square and Maximum Likelihood are the methods used by the linear and logistic regression algorithms, respectively, to determine the unknown parameters in the model.

OLS provides a minimum variance unbiased estimator if and only if the distribution of error terms in the regression is normal. This requirement is true for the linear regression, but not true for logistic regression. On the other hand, if the model is specified completely up to a finite number of unknown parameters, as is the case for logistic regression, then Maximum Likelihood estimation is asymptotically optimal. In that sense, what OLS is to linear regression, Maximum Likelihood is to logistic regression.

Answer 3.48: Without the bias term, the solution has to go through the origin. So, when all features are zero, the predicted value would also be zero. However, that may not be what the training data suggests. Adding a bias weight that does not depend on any of the features allows the hyperplane described by the learned weights to more easily fit the data.

Answer 3.49: Indeed, a generalized linear model could have nonlinear link and variance functions. However, the response is still a linear function of the random variables. Hence the term "linear" in its name.

Answer 3.50: The main drawback of GLM is that it is a parametric model that expects an optimal number of input terms of the right order. It can suffer precision losses if you didn't add the variables of right orders or the right interaction terms. To mitigate this situation, one often needs to use a regularization method (e.g., ridge, lasso, or least angles regression) to optimize instead of the simple Least Square.

Boosted decision trees [30] and random forest [25] are non-parametric modeling techniques so they don't suffer from the above drawback. However, these techniques are efficient for large datasets and may not provide better performance with small sample size.

Answer 3.51: A matrix is said to be ill-conditioned if it is singular and thus non-invertible. Computation of its inverse, or solution of a linear system of equations using it is prone to large numerical errors.

Answer 3.52: Regularization becomes necessary when the number of training samples is small or the model complexity is large. In these situations the model typically begins to ovefit or underfit the data. Regularization adds an extra cost term to the objective function in order to constrain the model parameters. This helps to reduce model complexity, so that the model can get better at predicting.

Answer 3.53: Regularization is a process of introducing additional information in regression in order to solve an ill-posed problem or to prevent overfitting. In general, a regularization term $R(f)$ is introduced in the loss function V that describes the cost of predicting $f(x)$ for label y

$$\min \sum_{i=1..n} V\left(f(x^i), y^i\right) + \lambda R(f) \tag{C.6}$$

$R(f)$ is typically a penalty on the complexity of f.

In the case of Lasso regression:

$$R(f) = \sum_j |w_j| = ||w||_1 = \ell_1\text{-norm of the weight vector} \tag{C.7}$$

For ridge regression:

$$R(f) = \sum_j w_j^2 = ||w||_2 = \ell_2\text{-norm of the weight vector} \tag{C.8}$$

So, for each λ, we have a solution. It controls the size of the coefficients and also the amount of regularization. As $\lambda \to 0$, we obtain the least squares solutions.

Answer 3.54: Ridge regression is defined by Eqs. C.6 and C.8. We need it as an additional constraint, via the $R(f)$ term in Eq. C.6, in order to solve ill-posed problems or to prevent overfitting. Such a constraint is absent in the OLS.

Answer 3.55: Lasso regression is defined by Eqs. C.6–C.7. It is different from ridge regression in the sense that the ℓ_1-norm of the weight vector defined in Eq. C.7 is different from the ℓ_2-norm of the weight vector of Eq. C.8. So, the constraints are different in the two cases. The OLS has no such constraint.

Answer 3.56: A simple rule of thumb is to use lasso regression when there are just a handful of features with medium–to–large sized correlations, and ridge regression in the case of a large number of features with small correlations [31].

In most practical situations, we typically have more than a handful of features and some degree of correlation among them. Therefore, ridge regression is almost always the preferred choice.

Answer 3.57: In the ridge regularization, the coefficients are normal distributed and in the lasso they are Laplace distributed. So, lasso is more likely to have one or more coefficients being zero, and therefore, easier to eliminate some useless input variables. Thus, lasso is more likely to result in sparse solutions.

Answer 3.58: For the same reason that lasso produces more sparse solutions, as explained in the answer to question 3.57.

Answer 3.59: Multi-collinearity (also called "collinearity") is a phenomenon in which two or more input variables in the regression model are highly correlated.

Answer 3.60: We can do a correlation analysis. If two input variables are highly correlated — let's say, correlation above 90%, although this threshold is subjective — then we can drop one. Alternatively, we can perform principal component analysis to reduce the number of features, or use a regularization scheme (e.g., ridge regression).

Answer 3.61: We can check for the presence of multi-collinearity in data and remove it using techniques described in the answer to question 3.60. Since at least one of the input features is redundant, we can try removing correlated variables from the model. If this has no impact on predictability, then we have a robust model.

On the other hand, if removing correlated variables leads to loss of information, then we need to keep them and use a regularization scheme, such as ridge or lasso regression, instead.

Answer 3.62: R^2 for a model is defined as

$$R^2 = \frac{\text{Explained variation}}{\text{Total variation}} = \frac{\sum_i (\text{data}_i - \text{mean})^2}{\sum_i (\text{data}_i - \text{prediction})^2} \tag{C.9}$$

Thus,

$$0 \leq R^2 \leq 1, \tag{C.10}$$

with zero indicating that the proposed model does not improve prediction over the mean model and one indicating perfect prediction. In general, the higher the R^2, the better the model fits the data.

Adjusted R^2 also indicates how well a model fits the data, but adjusts for the number of variables in the model.

$$R^2_{\text{adjusted}} = 1 - \left(1 - R^2\right) \cdot \left(\frac{n-1}{n-k-1}\right), \tag{C.11}$$

where n = number of data points, k = number of variables.

If we add useless variables to a model, adjusted R^2 will decrease. If we add useful variables, adjusted R^2 will increase.

$$R^2_{\text{adjusted}} \leq R^2 \tag{C.12}$$

Although both R^2 and the adjusted R^2 give us an idea of how well the model is doing, there is one main difference between them. R^2 assumes that every single variable explains the variation in the dependent variable. Adjusted R^2 tells us the percentage of variation explained by only the variables that actually affect the dependent variable.

Answer 3.63: We will consider adjusted R^2 as opposed to R^2 to evaluate the model's performance. The reason is that R^2 increases irrespective of improvement in prediction accuracy as we add more variables. But, adjusted R^2 would increase only if the additional variable improves the model's accuracy, otherwise stays same.

Tolerance, F, defined as

$$F = \frac{1}{\left(1 - R^2_{\text{adjusted}}\right)} \tag{C.13}$$

is an indicator of multi-collinearity, i.e., the fraction of variance in a predictor which cannot be accounted for by other predictors. Large values of tolerance is desirable.

Thus, larger values of adjusted R^2 and tolerance are preferable when comparing two models.

Answer 3.64: Using adjusted R^2 and F value.

Answer 3.65: See the answer to Question 3.63.

Answer 3.66: $R^2 = 1$ indicates that the proposed regression model fits the data perfectly.

Answer 3.67: An overfit model is one that is too complicated for the given data set. When this happens, the regression model is trying to

fit random fluctuations in the specific sample rather than reflecting the overall population. Overfitting can be avoided by using a simpler model with fewer parameters.

Answer 3.68: Since logistic regression is used to predict probabilities, we can use ROC curve, i.e., a plot of true positive vs. false positive, along with the rest of the confusion matrix elements to determine its performance.

Answer 3.69: Logistic regression is a technique to forecast the binary outcome from a linear combination of predictor variables.

Answer 3.70: In logistic regression, we predict not the actual probability but a transformed version of it, the "log odds". So instead of the probability $p(x)$, we deal with

$$\log\left(\frac{p(x)}{1-p(x)}\right) = b_0 + b_1\, x, \tag{C.14}$$

and find linear regression coefficients for the log odds. The first term b_0 is the intercept term, i.e., the value of the log odds when the predictor is equal to zero. The second term is the regression coefficient multiplied by some value of the predictor. Thus the decision boundary is linear in x.

Answer 3.71: A model is linear if its decision boundary is linear. For example, the decision boundary of a logistic regression is

$$\frac{1}{1+e^{-\sum_i w_i\, x_i}} = 0.5, \tag{C.15}$$

hence

$$e^{-\sum_i w_i\, x_i} = 1, \tag{C.16}$$

or

$$\sum_i w_i\, x_i = 0, \tag{C.17}$$

which is linear.

Answer 3.72: Yes, the logistic regression classifier is a linear classifier, because its decision boundary is linear. The classifier needs the inputs

to be linearly separable. However, decision trees and neural networks are NOT linear classifiers.

Answer 3.73: Logistic regression searches for a single linear decision boundary in the feature space, whereas decision tree partitions the feature space into two at various decision boundaries, thus resulting in nonlinear boundaries. Flexibility of decisions trees makes them prone to overfitting, hence the need for pruning. Logistic regression tends to be less susceptible to overfitting. Lastly, decision trees automatically take into account interactions terms like $x \cdot y$ between features x and y, but for logistic regression these need to be added manually.

The same comparison holds true for logistic regression vs. neural networks. The reason the decision boundary for a neural network is not linear is because there are at least two layers of sigmoid functions in the neural network.

Answer 3.74: Classification is the process of assigning data to one of a set of pre-determined class labels. It is a fundamental problem that has to be solved if machines are to make binary decisions.

Answer 3.75: Depending on one's perspective one can pick logistic regression, or Naïve Bayes, or even decision tree as the simplest classification algorithm. Pick one and describe it in detail in simple steps.

Answer 3.76: You can name a few classification algorithms you are familiar with, e.g., Naïve Bayes, neural networks, decision trees, SVMs, random forest, etc. Then describe which one you like most and why, e.g., due to their simplicity, speed, interpretability, or any other reason you may have.

Answer 3.77: Naïve Bayes is very simple since it assumes that each feature is independent of the others. If the independence assumption actually holds, a Naïve Bayes classifier will converge quicker than more complex models, so you need less training data. Even if the assumption doesn't hold, the classifier still often performs decently well in practice.

Answer 3.78: The Naïve Bayes classifier makes a very strong assumption that any two features are independent given the output class. Due to this, it disregards correlation between the features, hence, a "naïve" classifier.

Answer 3.79: Naïve Bayes is a conditional probability model based on Bayes' theorem with assumption of independence between features.

$$P(\text{Hypothesis}|\text{Data}) = \frac{P(\text{Hypothesis}) \times P(\text{Data}|\text{Hypothesis})}{P(\text{Data})} \tag{C.18}$$

Or, in plain English,

$$\text{posterior probability} = \frac{\text{prior belief} \times \text{likelihood}}{\text{marginal likelihood}} \tag{C.19}$$

A class's prior may be estimated from the training set or supplied externally. Maximum likelihood training is done by evaluating a closed-form expression for the likelihoods, which takes linear time, so can be quite fast.

Answer 3.80: One way to improve would be to de-correlate the features via PCA. Alternatively, we can try a model that works well with correlated features, e.g., decision trees.

Answer 3.81: By encoding the categorical values (via one-hot or label encoding) and converting them into numerical values.

Answer 3.82: Artificial neural networks, decision trees, random forest, support vector machines, etc. Table C.3 provides a summary of the main classification methods.

Answer 3.83: Linear Discriminant Analysis (LDA), also called Fisher's linear discriminant, is a method to find a linear combination of features that separates two or more classes.

If two classes have means $\vec{\mu}_1, \vec{\mu}_2$ and covariance Σ_1, Σ_2, then the linear combination of features $\vec{w} \cdot \vec{x}$ will have means $\vec{w} \cdot \vec{\mu}_i$ and variance

Classifier	Pros	Cons
Naïve Bayes	• Simple model • Fast • Easy to implement	• Over-simplification • Correlation among features ignored • High bias
Logistic Regression	• Discriminative model • Fast • Accuracy > than Naïve Bayes	• Slower than Naïve Bayes • Not so performant
Decision Trees	• Easily interpretable • Fast	• Prone to overfit
Random Forest	• Strong performance • Faster than ANN • Easy to implement	• Hyper-parameters to tune
ANN	• Powerful • Easy to use	• Slow • Less interpretable
SVM, kernels	• Powerful • Easily interpretable	• Slow • Choice of kernel not obvious
KNN	• No learning needed • Classes need not be linearly separable	• Sensitive to noise • Slow • Doesn't work well for large dimensions

Table C.3: Summary of classification methods.

$\vec{w}^T \Sigma_i \vec{w}$ for $i = 1, 2$. If this linear combination separates the two classes, then the vector $\vec{w}$ would be normal to the discriminant hyperplane. For example, the line that best divides two groups in a 2D plane is perpendicular to $\vec{w}$.

The separation between the two classes is the ratio of the variance between the classes to the variance within the classes

$$S = \frac{\sigma^2_{\text{between}}}{\sigma^2_{\text{within}}} = \frac{\left(\vec{w} \cdot (\vec{\mu}_2 - \vec{\mu}_1)\right)^2}{\vec{w}^T \left(\Sigma_1 + \Sigma_2\right) \vec{w}} \tag{C.20}$$

Maximum separation occurs when

$$\vec{w} \propto (\Sigma_1 + \Sigma_2)^{-1} (\vec{\mu}_2 - \vec{\mu}_1) \tag{C.21}$$

LDA is quite similar to PCA, however it uses the class label for dimensionality reduction. Thus it is a supervised learning algorithm.

PCA perform dimensionality reduction while preserving as much of the variance in the high dimensional space as possible. LDA, on the other hand, performs dimensionality reduction while preserving as much of the class discriminatory information as possible.

Answer 3.84: An artificial neural network is an interconnected group of processing elements, similar to the network of neurons in a brain. ANNs typically consist of multiple layers and the signal path traverses from front to back.

When training an ANN with a set of input and output data, we wish to adjust the weights in the input and hidden layers of ANN so that the ANN gives the same outputs as seen in the training data. The backpropagation algorithm looks for the minimum of the loss function by updating these weights.

Answer 3.85: We can think of logistic regression as a neural network with one input layer, one inner layer, and one output layer, where the sigmoid activation function acts in the hidden layer.

Answer 3.86: The activation function (also called "transfer function"), is a monotonically increasing, continuous, differentiable function, applied to the weighted input of a neuron to produce the final output.

The three most common choices for the activation function are

1. Sigmoid function:

$$f(x) = \frac{1}{1+e^x} \tag{C.22}$$

2. Hyperbolic tangent, or tanh, function

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{C.23}$$

3. Rectified linear unit (ReLU) function

$$f(x) = \max(0, x) \tag{C.24}$$

Answer 3.87: We need activation function for two basic reasons:

1. If there are no activation functions, the whole neural network would be reduced to a group of linear functions of the input. So, without activation functions, the neural network can't learn nonlinear relationships.
2. Each neuron in the neural network can be seen as recognizing a certain feature. Thus, activation $= 0$ indicates the absence of that feature, whereas activation $= 1$ indicates some contribution from the feature.

Answer 3.88: Because hyperbolic tangent function is antisymmetric. As explained in Ref [32], if the activation function is non-symmetric, as in the case of the sigmoid function, the output of of each neuron is restricted to the interval $[0, 1]$. Such a choice introduces a source of systematic bias for those neurons located beyond the first layer of the network.

To overcome this problem we need to use an antisymmetric activation function such as the hyperbolic tangent function. With this latter choice, the output of each neuron is permitted to assume both positive

and negative values in the interval [1, 1], in which case it is likely for its mean to be zero. If the network connectivity is large, backpropagation learning with antisymmetric activation functions can yield faster convergence than a similar process with non-symmetric activation functions.

Answer 3.89: Softmax regression (also called "multinomial logistic", "maximum entropy classifier", and "multi-class logistic regression") is a generalized form of logistic regression, which can be used in multi-class classification problems where the classes are mutually exclusive.

In logistic regression, as in Eq. C.15, we have two classes $y = 0$ and $y = 1$. Thus

$$p(y|x) = \frac{e^{(w)^T x}}{1 + e^{(w)^T x}} \tag{C.25}$$

Generalizing this to n classes $y^1, y^2, \ldots, y^n$, we have

$$p(y^i|x) = \frac{e^{(w^i)^T x}}{\sum_{i=1}^{n} e^{(w^i)^T x}} \tag{C.26}$$

The softmax function is often used in the final layer of neural networks, which are applied to classification problems.

Answer 3.90: They are not related in any meaningful sense. Sure, both can be used for regression. Neural networks is often used for regression and the Fourier transform is essentially a curve fit of multiple sine and cosine functions to some data.

Answer 3.91: The strong fluctuations could be due to very small batch size or a constant big learning rate. If I am using an adaptive step size for learning rate, then I just need to tune the batch size.

Answer 3.92: Deep Learning is a set of techniques to parameterize deep neural network structures, i.e., connected neural networks with many, many layers and parameters. Traditional neural networks

tended to overtrain in the case of large number of hidden layers, which are necessary for solving complicated problems, such as image recognition. Deep Learning finds optimal constants in the hidden layers so that overtraining can be minimized.

Deep Learning is different from traditional machine learning in the sense that it attempts to model high level abstractions in data. It also exploits the idea of hierarchical explanatory factors, where higher level — i.e., more abstract — concepts are learned from the lower level ones.

Answer 3.93: In a basic overview, they are just modifications of neural networks to work better with images (CNNs) and with text and language (RNNs). CNNs are several layers of neural network each with some number of nodes and convolution filters. RNNs are very similar to normal neural networks, except that one or more of the hidden layers is connected to itself. CNNs take a fixed size input and generate fixed-size outputs. RNNs can handle arbitrary input/output lengths, but require more data due to more complex model.

Answer 3.94: In a multi-layer RNN, backpropagated gradients for deeper layers are calculated as product of many gradients of activation functions. When those gradients are small, the product can easily vanish over time. For example, tanh derivative is < 1 for all inputs except 0. Sigmoid is even worse and is always ≤ 0.25. So, it becomes very hard to calculate and update.

Answer 3.95: One solution would be to use highly fine-tuned activation functions, which have "good" gradient values — non-zero over a large range, not too small, not too big. A more practical solution would be to use gating (i.e., pass = 1, block = 0), instead of the activation function. Then, the network can remember arbitrarily long input sequence, as long as those gates are all 1 along the path. This is how LSTM solves the vanishing gradient problem.

Answer 3.96: In LSTM, the activation function is the identity function with derivative equal to 1. So, the backpropagated gradient neither vanishes nor explodes, but remains constant. The effective weight is

equal to the forget gate activation. So, if the forget gate is on (activation close to 1.0), then the gradient does not vanish. That's why LSTM is so good at learning long range dependencies.

Answer 3.97: A kernel is a similarity function. It is a function that we provide to a machine learning algorithm. The function takes two inputs and spits out how similar they are. Kernel methods are a class of algorithms for pattern analysis, whose best known member is the support vector machine.

Answer 3.98: Kernel methods are different in the sense that they allow using linear classifier to solve non linear classification problems. We can think of kernel methods as moving the data to a higher dimensional space, where it becomes linearly separable.

Any linear model can be turned into a nonlinear model by applying the kernel trick to the model — replacing its features by a kernel function. We only need to compute the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates.

Answer 3.99: Some positives and negatives of the kernel methods are

- Positives
  - Can model nonlinear behavior.
  - Are non-parametric.
- Negatives
  - Have bad scaling with instances.
  - Need hyper-parameter tuning.

Answer 3.100: Quadratic for construction of the kernel matrix, cubic for matrix inversion.

Answer 3.101: Scaling issues can be overcome using some of the following ways:

- Low-rank kernel matrix approximations
- Selecting random features
- Approximations based on data near the query point
- Learning local structure of the data near the query point.

Answer 3.102: When training SVM, we optimize for the kernel parameters and the the cost of classification C. For kernel, a common choice is the Gaussian kernel, which has a single parameter γ.

- A large C, also known as "hard margin", gives low bias and high variance because it penalizes misclassification. Small C, also known as "soft margin", makes the cost of misclassification low, thus allowing more of them for the sake of wider cushion.
- A small γ gives pointed bump in the higher dimensions, hence low bias and high variance.

We find the best combination of C and γ using grid search (or, Bayesian optimization for better performance) with exponentially growing sequences of C and γ. For example,

$$C \in \{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}, \quad \gamma \in \{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\} \quad \text{(C.27)}$$

We perform cross-validation using each combination of parameter choices, and then pick the combination with the best cross-validation accuracy.

Answer 3.103: SVMs belong to a family of generalized linear classifiers that simultaneously minimize the classification error and maximize the geometric margin between the classes. Hence, they are known as maximum margin classifiers.

In SVM, the distance from the decision surface to the closest data point determines the margin of the classifier. Maximizing the margin is good, because points near the decision surface pose very uncertain classification decisions. The maximum margin classifier gives us safety margin. A slight error in measurement or a slight document variation will not cause misclassification.

Suppose our training dataset is $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$, where y_i are either 1 or -1. We want to find the maximum-margin hyperplane that divides classes $y_i = \pm 1$, such that the distance of the hyperplane from either class is maximized:

$$\begin{aligned} \vec{w} \cdot \vec{x} - b &\geq +1 \quad \text{if } y_i = +1, \text{ and} \\ \vec{w} \cdot \vec{x} - b &\leq -1 \quad \text{if } y_i = -1, \end{aligned} \tag{C.28}$$

where $\vec{w}$ is the normal vector to the hyperplane and $\frac{b}{\|\vec{w}\|}$ is its offset from the origin along $\vec{w}$. These constraints can be rewritten as:

$$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \quad \text{for all } i. \tag{C.29}$$

Thus, the maximum-margin hyperplane is completely determined by those $\vec{x}_i$ which lie nearest to it. These $\vec{x}_i$ are called support vectors.

Answer 3.104: See answer to the question 3.102.

Answer 3.105: Some popular choices for the kernel are

- Gaussian
$$e^{-\gamma |x - x'|^2} \tag{C.30}$$
- Linear
$$< x, x' > \tag{C.31}$$
- Polynomial
$$(\gamma < x, x' > + r)^d \tag{C.32}$$
- Sigmoid
$$\tanh(\gamma < x, x' > + r) \tag{C.33}$$

Since the kernel is a similarity measure, we choose a kernel that preserves the geometrical invariance of the problem. For example, the Gaussian kernel is invariant under rotation. Similarly, we can think of other kernels that are invariant under translation or some group representation. The choice of kernel can be automated using a cross-validation procedure.

Answer 3.106: The convex hull of a set X of points in the Euclidean space is the smallest convex set that contains X. For two points, this

is the straight line connecting the points. For three points, we get the triangle with the three points as corners.

In linearly separable data, convex hull represents the outer boundaries of the two groups. In the context of SVM, once we have two convex hulls, we get the maximum margin hyperplane as a perpendicular bisector between them. This hyperplane demarcates the largest separation between the two groups.

Answer 3.107: Here are some practical tips to consider:

- Support Vector Machine algorithms are not scale invariant, so it is very important to scale the training and test data. We need to scale each feature to $[0, 1]$ or $[-1, +1]$, or standardize it to have mean 0 and variance 1.
- The size of the kernel cache has a strong impact on the run time for large datasets. So, provided we have enough RAM available, we can increase the cache size to a higher value than the default, which is typically pretty low (e.g., 200 MB in *scikit-learn* [33]).

Answer 3.108: Decision Trees are a form of supervised learning method used for classification and regression. They are used to create models to predict target variable by learning simple decision rules inferred from the data features.

Answer 3.109: The main business reason is that the decision trees are simple to understand and interpret. They are also fast to train and easy to visualize.

Answer 3.110: We start with the entire data set as input. Then we look for a split that maximizes the separation of the classes. A split is any test on a feature that divides the data in two sets. Continue splitting on each set using remaining features until some stopping criteria are met. The resulting tree is the decision tree model.

Answer 3.111: Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits

the set of classes to reduce impurity. Popular impurity measures used in the decision tree splitting are Gini index and information gain.

Answer 3.112: The two most frequently used impurity measures are

$$\text{Gini index} = \sum_{i=1}^{J} f_i (1 - f_i), \tag{C.34}$$

where f_i is the fraction of items labeled with class i and J is the total number of classes.

$$\text{Information gain} = -\sum_{i=1}^{J} p_i \log_2(p_i), \tag{C.35}$$

where p_i is the fraction of each class present in the child node that results from the split.

Answer 3.113: No. A tree that is too large risks overfitting the training data and poorly generalizing to new samples. However, a small tree may not be able to capture all available information in data. So, some optimization in the growth of the tree is necessary.

Answer 3.114: Pruning is a technique to reduce the size of decision trees, by removing sections of the tree that provide little statistical power. Pruning reduces the complexity of the final classifier and improves predictive accuracy.

Answer 3.115: Pruning can be done top down or bottom up. In the top-down pruning, we apply some stopping criteria while growing the tree. In the bottom-up pruning, we replace bottom sub-tree by a leaf node. Class label of the resulting leaf node is same as the majority class of the sub-tree. If the prediction accuracy is not affected, then the change is kept.

Answer 3.116: Single decision tree models often overfit the data, making them poor predictive models[34]. An ensemble of trees built on samples of the data becomes a powerful predictor by averaging away the errors of the individual trees.

Answer 3.117: Ensemble learning is the process by which multiple classifier models are trained and combined to make prediction. For example, a single decision tree is a pretty weak classifier. But when multiple independent decision trees are trained on the same data and their outputs combined in an optimal way, the resultant classifier is quite strong. We use ensemble learning when component classifiers are more accurate than chance and are independent.

Answer 3.118: A random forest is an ensemble learning method that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy. The sub-sample size can be the same as the original input sample size or less, but the samples are drawn with replacement. Also, a particular number of features are selected at random to train on.

Answer 3.119: Some other ways to combine trees in an ensemble learning are *bagging* and *boosting*.

- In *boosting*, we draw a random subset of training samples without replacement to train a weak learner C_1. Draw second random training subset without replacement and add 50% (or, some algorithmically determined variable fraction) of the samples that were previously misclassified to train a weak learner C_2. If C_1 and C_2 disagree on a subset of the training data, then use that subset to train a third weak learner C_3. Combine all the weak learners either via majority vote or by taking their average.
- In *bagging*, we generate N different bootstrap training samples, where $N << N_{\text{Total}}$. Then we train algorithm on each sample separately and take the average prediction.

Answer 3.120: *Boosting* and *bagging* are both ensemble learning techniques, in which a number of weak classifiers are combined — through averaging or majority vote — to create a strong classifier. *Bagging* means that we bootstrap samples with replacement from the data and train a weak learner on each sample. In *boosting*, we use the entire data to train each learner, but instances that were misclassified by the previous learners are given more weight, so that subsequent learners give more focus to them during training.

Answer 3.121: The fundamental difference is, random forest uses randomized sampling (also called *bagging*) of data and features to reduce variance. Gradient boosting algorithms use *boosting* of event weights to achieve the same objective. In random forest, each record in the training data counts equally, while in GBM the record with larger error rate typically gets higher weight in the subsequent iteration of the training process.

Answer 3.122: The decision tree algorithm works top-down by choosing a variable at each step that best splits the node. Commonly used metrics to decide how to split nodes are Gini impurity (defined in Eq. C.34) and information gain (defined in Eq. C.35). The split that maximizes the metric is the best split.

Answer 3.123: Here are some practical tips to consider:

- Decision trees tend to overfit the data if there is a large number of features. Therefore, getting the right ratio of training samples to number of features is important.
- The number of samples required to populate a tree doubles for each additional depth. Therefore, we need to tightly control the depth of the decision tree. A simple rule of thumb is to start with depth $= 3$ and a sufficiently large value for the minimum number of samples at leaf nodes.
- In the case of unbalanced datasets, the decision tree can become biased toward the dominant classes. We can balance the training data by sampling an equal number from each class, or by normalizing the sum of the sample weights for each class to 1.

Answer 3.124: We first try to tune the number of features randomly selected at each decision point while building trees. The default setting is square root of the number of available features, which is not always the most optimal. Then, we try tuning the number of trees in the forest, the maximum tree depth, and the minimum number of samples required to split an internal node, etc.

Answer 3.125: Neither. Cross-validation on time-ordered data is a bit tricky. If some pattern emerges at a later time, then the training model

can still pick up on it, even though it wasn't part of the early dataset. Resampling the dataset will separate these trends, and we might end up validating on past years, which is incorrect. So, instead of naïve LOOCV, forward chaining would be a better procedure:

- fold 1 : training [1], test [2]
- fold 2 : training [1, 2], test [3]
- fold 3 : training [1, 2, 3], test [4] ... etc.,

where the labels $1, 2, 3, 4, \ldots$ represent the corresponding years.

Answer 3.126: Cluster analysis is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups. Clustering can be formulated as a multi-objective optimization problem. Some well-known clustering methods include K-means clustering, DBSCAN, agglomerative clustering, etc.

Answer 3.127: K-means is an algorithm to find K cluster centers in the data and assign the data points to the nearest cluster center, such that the squared distances from the cluster are minimized. Given a set of points, the algorithm aims to minimize the within-cluster sum of squares

$$\sum_{i=1}^{K} \sum_{x \text{ in } c_i} |x - c_i|^2, \tag{C.36}$$

where c_i is the mean of the points in cluster i.

The algorithm starts with K arbitrary cluster centers. It then assigns each data point to the cluster center whose distance from the point is minimum. The cluster centers are recomputed based on this assignment. The process is repeated until the assignments don't change.

Answer 3.128: Since K-means clustering is unsupervised, it's hard to know without context what the best clustering is. One can examine some post-clustering parameters, such as inter-cluster distance (i.e.,

the sum of the square distance between each pair of cluster centroids), maximum radius (i.e., the largest distance from a data point to its cluster centroid), etc. But in general, the choice of the quality metric would depend on the purpose of clustering.

Answer 3.129: An example of a good, fast clustering algorithm is *Hierarchical Agglomerative Clustering*, which has typical time complexity $O(n^2 \log n)$. A polynomial time complexity of low order should still be considered reasonable.

See answer to the above question regarding the quality of clustering.

Answer 3.130: The number of clusters, K, in the K-means algorithm generally depends on the problem at hand. For example, to separate a population into 3 shirt sizes, the obvious choice of K would be 3. In situations where we don't have a preferred value, one rule of thumb is to take the square root of the number of data points divided by two

$$K = \sqrt{\frac{N}{2}} \tag{C.37}$$

A more precise way is the *elbow method*. Here we plot the sum of squared error

$$\text{SSE} = \sum_{x \text{ in } c_i} |x - c_i|^2 \tag{C.38}$$

versus K, then choose the K at which the SSE decreases abruptly.

Answer 3.131: They are two totally different things, although both have the letter "k" in their acronym. kNN, or k-nearest neighbors, is a classification algorithm, where the symbol k is an integer describing the number of neighboring data points that influence the classification of a given observation. K-means is a clustering algorithm, where the symbol K is an integer describing the number of clusters to be created from the given data. Both accomplish different tasks.

Answer 3.132: The main reason is that the Manhattan distance has dimension restrictions. It calculates distance horizontally or vertically only. On the other hand, Euclidean distance can be used in any multi-dimensional space. In fact, in both K-means and kNN algorithms,

we need to compute the centroid of a cluster of points. The term "centroid" itself is from Euclidean geometry and refers to multivariate mean in the Euclidean space.

Also, the sum of squared deviations from centroid — which is what both algorithms try to minimize — is equal to the sum of pairwise squared Euclidean distances divided by the number of points. This nice property doesn't generally apply to non-Euclidean distances.

Answer 3.133: Instead of minimizing the squared distances from the nearest cluster-center, we can minimize for something that takes into account distance from other clusters. For example we can have an objective function like

$$\sum_{i=1}^{n}\sum_{j=1}^{c} w_{ij}\,|x_i - c_j|^2, \tag{C.39}$$

where

$$w_{ij} = \frac{1}{\sum_{k=1}^{c}\left(\frac{x_i - c_j}{x_i - c_k}\right)^2} \tag{C.40}$$

Answer 3.134: DBSCAN is a density-based clustering algorithm. Given a set of points, it groups together points that are closely packed together. The algorithm requires two parameters: the maximum neighborhood radius (ε) and the minimum number of points required to form a dense region. It starts with an arbitrary starting point. This point's ε-neighborhood is scanned, and if it contains sufficiently many points, a cluster is started. All points within the ε-neighborhood are added to the cluster, so are each point's own ε-neighborhood, if they also happen to be dense. This process continues until the density-connected cluster is completely found. Then, a new unvisited point is scanned and the whole processed is repeated until there are no unvisited points left.

Answer 3.135: The k-nearest neighbors regression is appropriate in the cases where the dataset under consideration has highly localized non-parametric features. For example, spectroscopy data and recommendation systems.

Answer 3.136: Deep Forest is an alternative to deep neural networks for solving complicated problems, such as image recognition. Instead of multiple hidden layers, Deep Forest uses decision tree ensemble approach to achieve similar levels of performance. Because of this tree-based approach, Deep Forest is easier for theoretical analysis.

In contrast to deep neural networks, which require large-scale training data, Deep Forest can work well even with small training data. Furthermore, deep neural networks require significant effort in hyper-parameter tuning. Deep Forest is easier and more scalable to train since good performance can be achieved by almost same settings of hyper-parameters. For more details and an example implementation of Deep Forest, see Ref. [35].

Answer 3.137: Markov model is a stochastic model used to make predictions for a randomly changing system, where the future states depend only on the current state of the system and not on the sequence of events that preceded it [36]. That is, given the present, the future does not depend on the past. This assumption is called *Markov property* and any process with this property is called a *Markov process*. A well-known example of Markov process is the Brownian motion. Two most common types of Markov models for an autonomous system are: Markov chain and Hidden Markov model.

Answer 3.138: Markov chain is the simplest type of Markov model. It models the state of a system with a random variable that changes through time. The next value of the variable depends on the current value, but it is conditionally independent of the previous values. An example of Markov chain is Markov Chain Monte Carlo, which uses stochastic methods of performing a random walk to correctly sample from the joint distribution of the state.

Applications of Markov chain include predicting lines of customers arriving at an airport, exchange rates of currencies, storage systems such as dams, and Google's PageRank algorithm.

Answer 3.139: A hidden Markov model is a Markov chain for which the state is only partially observable, but the output — dependent on

the state — is fully visible. In other words, observations are related to the state of the system, but they are typically insufficient to precisely determine the state. The adjective "hidden" refers to the state sequence through which the model passes, not to the parameters of the model. Since each state has a probability distribution over the possible output tokens, the sequence of tokens gives some information about the sequence of states.

Applications of hidden Markov model include pattern recognition in speech, handwriting, gesture, parts-of-speech tagging, etc.

Answer 3.140: Probabilistic graphical model (PGM) is a framework to encode joint probability distributions over complex domains and large numbers of random variables that interact with one another [37]. Generally, PGMs use a graph-based representation as the foundation for encoding a complete distribution over multi-dimensional space.

Two types of PGMs are commonly used:

1. Bayesian networks
2. Markov networks

Answer 3.141: Markov network is a sequence of random variables $x = x_1, x_2, \ldots, x_t$, which generally model a single concept like weather and its changes over time. So, it's really just one random variable, but with many copies of it across time. We are interested in the state of the variable after a very long time.

On the other hand, a Bayesian Network is a directed acyclic graph (DAG), which represents a factorization of joint probability distribution and conditional independence between some variables. Here we are interested in factorization and independence between variables to estimate the joint probability distribution.

Answer 3.142: "No Free Lunch" is a folklore theorem, which states that if an algorithm performs well on a certain class of problems, then it necessarily pays for that with degraded performance on the set of all remaining problems.

Answer 3.143: Splines and lowess are the methods to draw a curve through the data points. Unlike regression, they don't fit a model to the data and don't provide best-fit values of parameters. A spline curve goes through every data point, and may wiggle a lot to do so. Lowess does local regression to follow the trend of the data and allows to draw a smoother curve.

Answer 3.144: Adversarial machine learning is the process of explicitly training a model on adversarial dataset that contains features specifically designed to fool the model. For example, attack on spam filtering, where spam messages are obfuscated through misspelling of bad words or insertion of good words.

Generative adversarial networks are unsupervised learning method designed to work in adversarial environment. They are generally implemented as a system of two neural networks — one generative and the other discriminative — competing against each other in a zero-sum game. For more details see Refs. [38, 39].

Answer 3.145: CRISP-DM stands for Cross-industry Standard Process for Data Mining. Originally conceived in 1996, it is the most widely used form of data-mining model today because of its robustness and well-proven methodology. It includes descriptions of the typical phases of a project, the tasks involved with each phase, and an explanation of the relationships between these tasks.

CRISP-DM breaks the process of data mining into six major phases:

1. Business understanding
2. Data understanding
3. Data preparation
4. Modeling
5. Evaluation
6. Deployment

The sequence of the phases is not strict and can be customized easily. In fact, most projects move back and forth between phases as necessary.

Appendix D

Data Modeling Answers

This appendix contains answers to the questions posed in Chapter 4.

Answer 4.1: This is a typical conversation starter question. Keep an elevator pitch version ready!

Answer 4.2: Another classic question that is asked quite frequently. You get only a few minutes to both describe your experience with any of the mentioned ML tools — Scikit-learn [33], Weka [40], R [41], TMVA [42], Spark [43], TensorFlow [44], Torch [45], etc. — and quickly write down schematic pipeline for a simple example.

Answer 4.3: The interviewer expects you to write down each step explicitly [31], e.g.,

- Data acquisition: Including access to databases, distributed clusters, and context information such as labels and meta data
- Data cleaning and feature extraction: This may involve significant effort and may have its own mini-projects and sub-steps
- Feature selection: Compare feature distributions for each category/label to determine their statistical power, then decide which features to keep for further analysis
- Encoding of categorical variables: This is often necessary to include Boolean features in the machine learning model

- Dimensionality reduction: Via techniques such as PCA, ICA, SVD etc.
- First prototype model: This could be a simple model, such as Naïve Bayes or *K*-means clustering to establish baseline expectation
- Full blown model training: Typically a more complex model – often based on neural nets or ensemble learning — that gives better performance than the prototype model
- Check for overtraining: Model's performance as a function of the training cycles and number of features
- Validation/cross-validation procedures
- Performance study: Recall vs. false positive
- Operating point selection: This is often based on business needs, e.g., to either improve recall or reduce false positive
- Optimization and deployment: Including testing and integration into the production code

After explaining the above steps, you may be asked to estimate how much time — absolute or percentage-wise — you would spend on each step.

Answer 4.4: This is somewhat domain specific, but keep a simple version ready for the purpose of story telling. Table D.1 lists some generic problem types and what kind of ML technique is applicable to them.

Answer 4.5: The interviewer expects to see a workflow for the example cleaning procedure. For example,

- Data parsing, e.g., detection of syntax errors, misspellings, etc.
- Transformation, e.g., value/type conversions
- Duplicate elimination
- Treatment of missing data, etc.

Problem	Supervised Learning	Unsupervised Learning
Segmentation	Decision Trees, Markov Chain	*K*-means clustering
Concept description	Naïve Bayes	Concept clustering, Word embedding, taxonomy, LSI, LDA
Classification, categorization	Naïve Bayes, Logistic Regression, Decision Trees, Random Forest, SVM, Neural Nets	*k*-nearest Neighbors, *K*-means clustering, DBSCAN
Prediction, regression analysis	Linear Regression, Decision Trees, Random Forest, AdaBoost, SVM, Neural Nets	*k*-nearest Neighbors
Customer id & analysis	Classification algorithms mentioned above	Named entity recognition, *K*-means clustering,
Dependency analysis	Decision Trees, Markov Chain, Bayesian networks	PCA, graph database
Recommender systems	Bayesian Networks, Hidden Markov Models	*k*-nearest neighbors

Table D.1: Machine Learning techniques per problem type.

Answer 4.6: No. It simply means that the climate change is correlated with decrease in number of pirates. Correlation doesn't necessarily imply causation.

Answer 4.7: This is possible if X and Z are highly correlated.

Answer 4.8: Since our visual imagination is not very intuitive beyond 3 dimensions, to display higher dimensional data we need to adjust either the visualization or the data. We can show multiple 2D scatter plots or 3D contours by projecting out other dimensions. We could show all 5 dimensions using a parallel coordinates plot.

Answer 4.9: We need to do A/B testing for 9 experiments (3 sizes $\times$ 3 positions). In each case, we want to have small enough uncertainty, or large enough sample size, so that we can make claim with 95% confidence. In other words, $1.96 \times \sqrt{N}$ should be smaller than the difference in click-rates between any two experiments.

In the extreme case, if one experiment gets no click, then the other experiment needs to have at least 4 clicks to claim better performance. However, in the case of null hypothesis (i.e., all experiments doing equally well), no sample size may be large enough to provide 95% confidence for one experiment doing better than others.

Answer 4.10: This is a regression problem. We aim to predict the amount of milk produced in a day (y) as a linear function of input features (x_i):

$$y = w_0 + \sum_i w_i x_i, \tag{D.1}$$

where the input features are temperature, humidity, and feed consumption. We may also want to normalize the features

$$\text{feature} \rightarrow \frac{(\text{value} - \text{min\_value})}{(\text{max\_value} - \text{min\_value})} \tag{D.2}$$

and perform a principal component analysis. If we have just a handful of data points, then we may also want to add a regularization term, such as $\lambda \sum_i w_i^2$ or $\lambda \sum_i |w_i|$ in the regression Eq. D.1.

Answer 4.11: There are two components to this problem:

- How to process high dimensional data on a limited memory machine?
- How to reduce dimensionality of a high dimensional dataset?

To address the first part, you could suggest randomly sampling a small fraction of data, then running analysis on it.

As for the second part, we can use Principal Component Analysis to reduce dimensions in the case of numerical data. For text, we can try a reduction technique, such as Latent Semantic Indexing. In addition, we will likely need to do feature selection, by algorithmically finding out which columns are more significant, thus reducing the dimensions that way.

Answer 4.12: This is a classification problem, where the output can have five possible values. The number of raw input dimensions is $1920 \times 1080 \times 3$. The input dimensions can be reduced by employing principal component analysis of pixel-wise covariances and keeping only the first N principal components, where the value of N (likely in the range $10 - 200$) can be optimized for prediction accuracy and verified on validation data.

Since there are five possible values for output, we can use one-hot encoding to label our data: [10000], [01000], [00100], [00010], [00001], and optimize for cross-entropy loss. Often one uses softmax distribution to compute the probability of each output class for a given set of inputs.

Answer 4.13: We can start with a basic model for the average annual temperature (y) containing three terms:

1. Global warming of w_1° C per year,
2. A natural cycle (cosine function) with amplitude w_2 and period w_3, and
3. A random noise of amplitude w_0.

Thus,

$$y = w_0 + w_1 x + w_2 \cos\left(\frac{2\pi}{w_3} x\right), \tag{D.3}$$

where x is the number of years since the beginning of data-taking. If this model turns out to be inadequate to explain the data then we can progressively add more complexity, e.g., by adding more cyclic terms and step functions for the El Niño effect.

Answer 4.14: This is an LDA problem. We can start with headlines from around the world and group them into topics. Then, we use LDA to compare a document to two topics and determining which topic is closer to the document, across all combinations of topics which seem broadly relevant to a story. Finally, we aggregate all relevant information together into an article.

Answer 4.15: This question has two components:

- Minimizing the cost of delivering food to save money, and
- Optimizing delivery routes in order to deliver food on time.

The first is a regression problem, where we need to optimize for two competing costs — one incurred on better on-time delivery and the other due to free food vouchers. The second is a route optimization problem.

Answer 4.16: The basic idea for this kind of recommendation system comes from collaborative filtering. The collaborative filtering algorithm considers user behavior for recommending items. It uses other users' behavior and preference of items to recommend similar items to new users. Typically, features of the items are not known.

In particular, the Amazon recommender system is based on *item-to-item* collaborative filtering. For each item X, Amazon builds a neighborhood of related items $S(X)$. Whenever a customer buys/views an item, Amazon recommends new items from that item's neighborhood.

Answer 4.17: Dimension reduction is the process of reducing the number of random variables under consideration, via obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

Answer 4.18: The curse of dimensionality refers to various phenomena that arise when analyzing data in high-dimensional spaces that do not occur in the three-dimensional physical space of everyday experience. Due to the curse of dimensionality, machine learning algorithms may perform poorly in high-dimensional data. We deal with it via dimension reduction and using algorithms that are more robust in high dimensions.

Answer 4.19: Having large feature space means that the model will be analyzing data in high-dimensional spaces. This can cause the model to suffer from curse of dimensionality. One bad consequence of this could be overfitting, i.e., the model optimizing for random fluctuation in the finite training set. Another disadvantage of having too many features is that the model may take a very long time to train. Run-time complexity of most algorithms grows exponentially with the increase in dimension.

Answer 4.20: In the context of data analysis, eigenvectors are a set of basis coordinates in which the correlation matrix becomes diagonal, thus allowing the independent variables defined in this coordinate system to be uncorrelated. The eigenvalues represent data variance along each new basis. Eigenvectors are used to reduce the dimension of large data sets. An important application of eigenfunctions is the PCA.

Answer 4.21: Principal Component Analysis is a procedure to convert a set of observations of correlated variables into a set of linearly uncorrelated variables, called principal components. The goal of the PCA is to explain the maximum amount of variance with the fewest number of principal components. Thus, it finds a linear projection of high dimensional data into a lower dimensional subspace. By reducing the dimensionality of the problem, it helps in reducing overfitting and run-time complexity of the models used to describe the data.

Answer 4.22: PCA is equivalent to finding the eigenvalues and eigenvectors of the covariance/correlation matrix. Therefore, eigenvalue decomposition technique can be used to compute principal components.

Answer 4.23: Independent Component Analysis is a procedure to convert a set of observations of correlated variables into a set of independent components. This is done by maximizing the absolute value of normalized kurtosis — a 4^{th} order statistic:

$$\max \quad \frac{1}{N}\sum_i a'(x_i - \bar{x})^4. \tag{D.4}$$

PCA, on the other hand, aims to maximize variance for linearly uncorrelated principal components.

Answer 4.24: Yes, the PCA is a linear transformation, because each principal component is a linear combination of the original features.

Answer 4.25: The connection between PCA and SVD is that the covariance matrix in PCA can be diagonalized using SVD:

$$M^T M = VL^2V, \tag{D.5}$$

where M is the covariance matrix, V is an orthogonal matrix of the the "right singular-vectors" of M, and L^2 is a diagonal matrix containing information about the scaling part of SVD. An alternative way to diagonalize the covariance matrix is via eigenvalue decomposition. Generally, SVD is preferable over EVD because the extra step of calculating $M^T M$ in EVD can lead to numerical rounding errors.

Answer 4.26: Yes, we should remove the correlated variables first because, in the presence of correlated variables, the variance explained by a particular component gets inflated. Then we can run PCA, which will give us a minimum set of variables to represent the data. If we need further dimension reduction (e.g., due to space or memory constraints), then we will algorithmically find out the least significant features and drop them.

Answer 4.27: PCA is most effective when there are few uncorrelated principal components, along which the data show high variance.

Answer 4.28: Projecting data on a random vector is not a valid dimensionality reduction technique. However, if the vector is chosen in

such a way that it can explain most of the data variance along that direction and is uncorrelated with other features, then it could be a valid technique.

Answer 4.29: PCA is based on the covariance matrix, which is derived from centered data. Hence, we need to center the data for PCA. If we don't do centering, the PCA applied on it may not be very useful as the eigenvectors would look quite different than what they should be.

Answer 4.30: While performing PCA, we need to scale the features in data to have unit standard deviations, especially if they have different units. Otherwise, each feature will end up with the same weight.

Answer 4.31: Yes, rotation (orthogonal) is often necessary. The goal of PCA is to select fewer components than the features that can explain most of the variance in data. In order to maximize the difference between variance captured by the components, we need to rotate.

If we don't rotate the components, the effect of PCA will diminish. PCA without rotation could still provide a set of orthogonal, uncorrelated components. However, the components would be uninterpretable and we'll have to select more number of components to explain variance in the data.

Answer 4.32: A data sample is dimensionally sparse if it has low probability of having at least one row for every combination of feature dimensions. The sample is density-sparse if the fraction of cells (rows $\times$ columns) with non-zero/non-default values is small.

Answer 4.33: Some useful dimensionality reduction steps during data preprocessing are:

- Remove features with too many missing values.
- Filter out features that have constant value or very low variance.
- Reduce highly correlated columns.

Answer 4.34: For any model building exercise, we want to keep a robust set of features and exclude the noisy ones that don't have enough statistical power. There are a few rules of thumb to accomplish this. First, use domain knowledge, whenever available, to construct a better set of features. Second, normalize the features so that their values are comparable to 1. Third, compute correlation between each pair of variables and remove any variable that is $\approx 100\%$ correlated with another variable. Finally, use a variable ranking method, such as those based on information gain or linear regression.

Answer 4.35: Feature engineering is the process of using insights derived from the context of a problem and expert domain knowledge to create useful features from data. These features are important in developing highly predictive models[34]. Feature engineering is fundamental to the application of machine learning.

We need feature engineering in order to identify relevant features that can be ingested by machine learning algorithms to provide accurate predictions.

Answer 4.36: No. Their predictive power and variation could be quite different from one another.

Answer 4.37: If there are too few variables, then the model may not have enough flexibility to be able to fit the data. This would result in a biased model. If there are too many variables, then there is always the danger of overfitting the data by optimizing for random variations. This would result in a model with large variance, thus making it less useful for predicting on unknown data sets.

Answer 4.38: We should use the minimum number of features that capture the maximum available information in the data. To select the best features, we can use feature engineering, PCA, remove redundant and low variance features, etc.

Answer 4.39: We need to encode categorical variables into numerical values. There are two ways to do this — one-hot encoding and label encoding. In one-hot encoding, we generate one Boolean feature for

each category. Only one of these features can take on the value 1 for each sample. For example, {"red": 100, "blue": 010, "green": 001}. In label encoding, we simply assign a numerical value to each categorical value, e.g., {"red": 1, "blue": 2, "green": 3}, thus keeping the number of features intact. One-hot encoding works better with most machine learning algorithms. However, some algorithms like decision trees, can handle categorical values natively. In such cases, one-hot encoding is not necessary.

Answer 4.40: Using one-hot encoding will result in an increase in the number of features, because it creates a new feature for each label present in the categorical variable. For example, let's say we have a categorical variable *color* with three labels "red", "blue", and "green". Its one-hot encoding will generate three features *color.red*, *color.blue*, and *color.green*, each with value 0 or 1. There is equal Hamming distance between any two of these three features.

In label encoding, the labels of a categorical variable get encoded as 0 through N, where N is the total number of labels present. Taking the above example, the label encoding of the categorical variable *color* will result in just one feature with three possible values: {"red": 1, "blue": 2, "green": 3}. So, no new feature is created. However, there is one drawback to label encoding, it adds misleading information. There is no reason why red is more similar to blue than to green. For this reason, using one-hot encoding, followed by some dimensionality reduction — like PCA, would probably be a better solution.

Answer 4.41: In this case we can one-hot encode a subset of the unique feature values, i.e., the ones that appear with more than some threshold frequency. Then, either we drop the remaining ones or encode them as a special value.

Answer 4.42: If there is no pattern in the missing data and the overall sample is large enough, then we can drop them without loss of statistical power. However, if there is a pattern, then we need to replace missing values either by contacting the source of the data, or by some average imputation, or through regression [46]. The exact course of action would depend on the task at hand.

Answer 4.43: Since data is spread across median, we can assume it is a normal distribution. In a normal distribution, 68% of the data lies within 1 standard deviation from the mean, and 32% outside of 1 standard deviation. Therefore, in this case, 32% of the data would remain unaffected by the missing values.

Answer 4.44: See the answer to question 4.42. Here 8 of the 50 variables have some missing data. If missing data happen to occur simultaneously for multiple variables, then we can drop those data records. If a record contains just one missing value, then we can try imputation via averaging or regression to see if either technique performs better than simply dropping the missing data.

Answer 4.45: Some popular feature selection methods are:

- Filter methods — like PCA, correlation analysis, etc.
- Linear discriminant analysis.
- ANOVA, which aims to select the best uncorrelated features.

These methods are independent of the choice of the machine learning model. In addition, there are model-dependent wrapper methods that use a subset of features to train the model. Based on the inferences, we decide to add or remove features. Embedded methods, like Lasso and Ridge regressions, have their own built-in feature selection methods which penalizes some features to reduce overfitting.

Answer 4.46: If the outliers in the data are due to measurement error, then we should either filter them out or use an algorithm that is robust against outliers. This is because some models are more sensitive to outliers than others. For instance, AdaBoost may put large weights on outliers, while decision tree might simply count each outlier as one mis-classification. Often the most predictive models are quite sensitive to outliers, so removing them from the get go is generally a good idea[34].

Answer 4.47: There is no single silver bullet for handling an imbalanced dataset during the training. So, we need to try several approaches and choose the one that gives the best performance. One

approach would be to under-sample the over-represented class. Alternatively, we can try bootstrapping the under-represented class.

Yet another way would be to generate synthetic samples (simulations) of the under-represented class [47]. We can also try penalized classification, which imposes an additional cost on the model for misclassifying the minority class. Finally, some learning algorithms perform better than others in the presence of unbalanced dataset, so we can try a few different algorithms [48].

Answer 4.48: Data sparsity can be a problem in some machine learning models, such as k-nearest-neighbors and recommender systems. The model is limited to recommending items that are similar to a given set of items. If that set is small, and the number of users and items is large, the sparsity makes most recommendations quite unrelated to the user's items. As a result, new users have trouble getting recommendations, while at the same time brand-new items have trouble making it into any user's recommendations.

One way to deal with sparsity is by using latent-factor models. Here we model each user-item interaction as a combination of some small number of factors; then correlate these factors with available items to make more meaningful predictions.

Answer 4.49: Black and white lists are two ways of filtering data. A white-list is a list of things that are known to be good. The opposite is a blacklist, i.e., a list of things known to be bad. In the context of spam filter, we can have a white-list of people we are willing to receive email from without checking for spam, and a blacklist of known spammers whose emails get automatically flagged as spam.

Answer 4.50: Staying with the example of spam detection, we can have a blacklist of known spammers and a white-list of verified email addresses. Emails from anyone not on either list can be examined using a set of positive rules (or, Bayesian filter) to determine whether they look like a spam or not.

Answer 4.51: We can augment our data using a number of random transformations, so that the classifier model would never see twice

the exact same image. These transformations include randomly rotating the image, translating vertically or horizontally, rescaling the RGB coefficients, shearing transformations, zooming in or out, horizontal flips, etc.

Answer 4.52: We can design the test such that a handful of questions are asked in two different ways. If the answers to these different ways of asking the same questions vary wildly, then that would indicate random response.

Answer 4.53: Overfitting refers to the situation when a statistical model describes random noise instead of the underlying relationship. This typically occurs when the model has too many parameters compared to the number of observations. Therefore, we can avoid overfitting by making the model simpler, i.e., by reducing the number of free parameters in the model.

Answer 4.54: There is a tradeoff between a model's ability to minimize bias and variance. A high bias means we have an underperforming model, perhaps too simple, which keeps on missing important trends in the data. A high variance means that our model is too complex that overfits the training data and performs poorly on test data. In general, a more complex model ends up giving less bias, but a higher variance. Hence, an optimized trade-off between the two is fundamental to machine learning.

Answer 4.55: To check overfitting, we can compare error-rates in the training and validation sets as a function of the model complexity. Alternatively, we can do cross-validation by successively training and testing on subsets of data. In the case of small dataset, we can even bootstrap to form training and validation sets. Then, we compare error rates as a function of the number of model parameters and also as function of the number of training cycles.

Answer 4.56: In addition to classification accuracy, some other metrics to examine are the false positive and false negative rates, normalized residuals (χ^2) versus predicted values and versus each input feature, etc.

Answer 4.57: Mean Squared Error (MSE) is by far the most common measure of model performance. It is simply the average of the squares of the differences between the predicted and actual values. One problem with MSE is that it overemphasizes the importance of larger errors, even though the error might actually be very small in percentage. In addition to MSE, I would also look at mean absolute percentage error. For the purpose of diagnostics, I would analyze plots of actual and predicted values, residuals versus time and versus predicted values, correlation among input variables, and tests for overtraining.

Answer 4.58: In addition to classification accuracy, I also need to examine the false positive and false negative rates. Because it matters immensely what percentage of actual cancer patients my algorithm failed to detect and what percentage of non-cancer patients it ended up mis-identifying.

To illustrate the point, let's assume a population of 1000, out of which 1% has cancer. Our classification model has 96% accuracy. Then

$$\begin{aligned}
\text{Total population} &= 1000 \\
\text{Actual number of cancer patients} &= 0.01 \times 1000 = 10 \\
\text{Cancer patients detected by model} &= 0.96 \times 10 + 0.04 \times 990 = 49.2 \\
\text{True positives} &= 0.96 \times 10 = 9.6 \\
\text{False positives} &= 0.04 \times 990 = 39.6 \\
\text{False negatives} &= 0.04 \times 10 = 0.4
\end{aligned}$$

So, we have a model whose false positive is about 4 times higher than its true positive. In other words, it is pretty useless in predicting the number of cancer patients!

This is a classic case of imbalanced classes, where the overall classification accuracy is not a good metric of performance. The accuracy of the minority class (i.e., cancer patients) is what we are interested in. Hence, we need to closely examine the false positive and false negative rates.

Answer 4.59: Very likely the five models were highly correlated, as they used the same training data and same set of features. Ensem-

ble learning can provide superior result by combining several weak learners to create a strong learner. However, this approach works only when the individual models are uncorrelated.

Answer 4.60: This would happen if the unseen data have different distribution of input variables than the training data, i.e., if the training and unseen data are statistically different.

A likely reason for poor test accuracy in this particular case could be the random sampling of training data. Random sampling doesn't take into consideration the proportion of target classes. We can check this possibility by retraining the model after sampling the training data in a way that maintains the target distribution.

Answer 4.61: Precision and recall are defined as

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}} \tag{D.6}$$

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}} \tag{D.7}$$

ROC curve is a plot of one vs. the other (or, sometimes one versus (1 – other)).

Answer 4.62: The simplest thing would be to store the result in a file on disk if we have have access to the disk. Alternatively, we would need to format the result appropriately, then store in a database. The advantage of the latter approach is that we will have different versions of the record, so we can also monitor the evolution of the record over time.

Answer 4.63: We can conduct degradation test by analyzing variation in the model response in test data over time. In particular, we would check whether any observed degradation during the test period is statistically significant and assess its effect on the recall and false positive rates. If degradation over time is an expected behavior, then we can attempt to model it by a function of time and some possibly multidimensional random variables.

Answer 4.64: Here we should follow the Occam's Razor principle and choose the simpler of the two models, i.e., the one with fewer parameters.

Answer 4.65: Cross-validation is a model evaluation method for assessing how the results will generalize to an independent data set. The hold-out method is the simplest kind of cross-validation, in which the dataset is separated into two subsets, called the *training set* and the *test set*. Trained results obtained from the training set is applied on the test set to evaluate the model performance.

Answer 4.66: In 10-fold cross-validation, we break data into sets of size $N/10$. Then, we train on 9 of these and test on the remaining set. We repeat this 10 times with permutations of the training and test sets. We take mean accuracy as the performance metric.

Answer 4.67: In the hold-out method, we divide data into just two sets, then use one for training and the other for validation. In 10-fold CV, we divide data into 10 sets, then train on 9 sets and test on the remaining set iteratively.

Answer 4.68: By doing cross-validation.

Answer 4.69: We can select the best model based on their performance on an independent test dataset specifically set aside for this purpose.

Answer 4.70: Often models with high predictive power are quite complex and not very interpretable. However, in real life the model needs to be explained to a larger audience via simple story-telling, e.g., which features are used, how the algorithm works, what is the strategy, etc. Also, the models need to be easy to deploy and monitor by others. So, it is desirable to have a model that is powerful enough and interpretable. The exact trade-off between the two is mostly a business decision, using cost vs. benefit optimization.

Answer 4.71: Sensitivity analysis is a way to determine how changing an independent feature impacts the model prediction. High sensitivity means that the prediction depends strongly on a given feature.

Often high sensitivity features also provide high predictive power. However, under a slightly different feature distribution the model may perform much worse, thus making the model less robust. In the ideal world, we would like to have high sensitivity and high robustness, but in real life robustness is more desirable even at the cost of some predictability loss.

Answer 4.72: In the case where we expect test data to be noisier than the training data, it would be reasonable to train models with various levels of noise injected to test their robustness. However, the nature and relative amount of the injected noise must match closely to what is expected in the test data. A through cross-validation would be necessary to make sure that this approach actually works.

Answer 4.73: Let's assume we have a previously trained model and we just got a new training data set. In practice, most of the time we do not have enough new data to retrain from scratch, but maybe enough to fine-tune a pre-trained model. We can focus on a few specific parameters and scan them using grid search. Then minimize the cost function. A more robust way would be to perform stochastic gradient descent for this purpose.

Answer 4.74: See the answer to problem 4.73. In order to find the most optimal parameter values, especially in the case of small dataset, we can start with a pre-trained model or an approximate analytic solution. From thereon, we can either use iterative grid search, Bayesian optimization, or adaptive stochastic gradient descent to improve the model.

Answer 4.75: By model performance we generally mean some metric, such as MSE, that quantifies the difference between the predicted and actual values. On the other hand, model accuracy tells us how well the model fits a new unseen dataset, i.e., whether the model is suitable enough to solve the kind of problem it was designed for.

Model accuracy can be checked by analyzing the false positive and false negative rates, in addition to the accuracy. Clearly, it is desirable to have a model that is both accurate and well performant. It

can be argued that accuracy is more important because an arbitrarily precise model that doesn't fit new data is useless. However, an accurate but imprecise model can still be useful in combination with other information.

Answer 4.76: Amazon provides data on book's sales rank versus copies sold per week. The average number of copies sold decreases roughly as inverse of some polynomial function of the book's sales rank. We can start with such an analytic model and determine the parameters of this model by fitting to data. Then we will be able to estimate sales figure for any book, given its rank.

Answer 4.77: One hypothesis is that the spike in picture uploads is due to Halloween. This can be tested by comparing upload trends in the neighboring countries that do not observe Halloween. One can come up with other hypotheses and corresponding tests.

Answer 4.78: Yes, we can! We can define a new column for each data record using the formula

$$L = w_0 + w_1 X_1 + w_2 X_2 + ... + w_n X_n, \tag{D.8}$$

starting with initial guess values for the w's. Similarly, we can compute

$$P(x) = \frac{e^L}{1 + e^L} \tag{D.9}$$

and log likelihood

$$y \ln P(X) + (1 - y) \ln (1 - P(X)) \tag{D.10}$$

for each data record. Finally, we compute the sum of log likelihoods of all data records. The objective of logistic regression is to find the coefficients that maximize this sum. We will have to do this iteratively by changing the values of the w's in steps. Excel has numerical precision issues, but it can get close to an optimal solution.

Answer 4.79: We can start with some simple steps, such as eliminating fake reviews created by bogus accounts to game the ratings by, e.g., blacklisting of keywords. We can give more weight to trusted users.

We can flag all reviews that are almost identical and come from the same user.

Next, we can exclude prolific users who like everything or hate everything. We can also try to eliminate competitor reviews (e.g., two similar businesses in the same zip code bad-mouthing each other) and disgruntled employees bad-mouthing their former employer. If a business has received very few reviews in the past, but suddenly gets multiple similar comments within a day, then we may want to treat that as a special case that needs closer examination.

Answer 4.80: Number of colleague's LinkedIn connections can be estimated using

$$N_{\text{colleague}} = \left(\frac{N_{\text{shared}}}{N_{\text{my}}}\right) \times N_{\text{network}}, \tag{D.11}$$

where N_{shared} is the number of LinkedIn connections that are common to both me and my colleague and N_{my} is the number of my connections. The size of my overall LinkedIn network can be estimated using

$$N_{\text{network}} = N_{\text{2nd\_degree}} \times \frac{N_{\text{my}}}{\overline{N}_{\text{shared}}}, \tag{D.12}$$

where $N_{\text{2nd\_degree}}$ is the number of my 2nd degree connections and $\overline{N}$ denotes the average value.

Answer 4.81: There are many problems when dealing with small data, but they mainly revolve around high variance. For one, random fluctuations are high, so over-fitting becomes much harder to avoid. One ends up not only over-fitting the training data, but sometimes even the validation set. Outliers become hard to define and even harder to detect. Hypothesis tests need careful design to have any sensitivity.

Answer 4.82: The amount of data necessary to train a model depends largely on the nature of the prediction one is trying to make. A simple rule of thumb is that the training data should be at least roughly $10X$ the number of model parameters.

However, we can train a model with the amount of data in hand and evaluate its test error. If the test error is larger than we can live with, then we need to collect more data.

Answer 4.83: If we start with a random initialization of input variables and then run stochastic gradient descent to make them converge, we have no way of knowing if the solution is a local optimum or the global. So, if the training procedure converges at all, we may end up in a local optimum. This will not always result in the best possible solution.

However, we can run gradient descent several times with different random initializations. If the solution always converges to the same optimum, chances are that we have found the global optimum and not a local one.

Answer 4.84: This is an open-ended question that provides a good opportunity to showcase your data wrangling skills. Start enumerating how you will pre-process both numerical and text data, deal with missing information, remove duplicates and unprintable texts, and encode categorical variables, etc. Then describe how you may be able to use some property of data to simplify some of these tasks.

Answer 4.85: Just pick one of the provided examples, say topic modeling on email data, then dive deeper. Outline the data acquisition, pre-processing, model building, and prediction pipeline. Sprinkle anecdotes from your current or previous jobs to make the story interesting. This is another question that may invite follow ups.

Answer 4.86: Good visualizations:

- Simplicity of the plot, properly defined axis labels, and carefully chosen titles make digesting the data easy.
- The use of proportionally sized markers makes display fairly easy on the eyes.
- Adding trend lines to a scatter plot helps the reader by highlighting trends.

- Heat map with a single color and varying shades show the intensity better than multiple competing colors.

Bad visualizations:

- Use of too many dashed and dotted lines to display multiple graphs on the same plot can be distracting.
- Abundant use of red, blue, and orange colors in an stacked bar chart can mislead the reader into thinking that the colors mean good (blue) versus bad (orange and red) whereas these are just associated with a specific segment.
- A poorly designed pie chart with lots of items can make it difficult to make comparison between items when area is used for scaling.
- Showing a lot of data points, none of which identifies the most important trends that users need to pay attention to, is certainly a poor visualization.

Answer 4.87: Data visualization is an important part of data analysis. There are several excellent visualization tools available today to present the data as graphs, charts, and 3D models. Some popular ones are Matplotlib, ggplot, Tableau, D3.js, Plotly, etc. You can talk about the ones you use most regularly.

Answer 4.88: A data-driven recommender system recommends new items based on user-provided context attributes. Typically, such systems work in one of the following two ways:

- Collaborative filtering: To recommend an item that is likely to be bought by similar users, we analyze customer transaction history to find similar users, and then provide recommendations based on their preferences.
- Similarity-based: To recommend a particular category of items, e.g., a specific music genre or style of clothing, we can suggest new items similar to the ones the user bought or liked in the past. For this purpose, items may be matched based on visual or style similarity or similar tags, often informed by some kind of A/B testing.

To make predictions, the recommender systems typically use some variation of the k-nearest-neighbors classifier. The metric to define closeness could be classical Euclidean distance or the cosine distance, depending on the context.

In the case of large training data, one can even attempt a supervised learning model to provide recommendations for the items with the highest score. The most common ML algorithms used for such tasks are decision-trees and random forest as they are not only relatively efficient, but also easy to understand once the model is trained.

Answer 4.89: Some of the limitations of the Data-driven recommender systems are

- Cold start problem: When new items are added to the system, they first need to be liked/bought by a substantial number of users before they could be recommended to other users with similar tastes. Similarly, new users will need to like/buy a sufficiently large number of items to let the system capture their preferences accurately.
- Scalability: Most ML algorithms have polynomial or even exponential time complexity. As the numbers of users and items grow, this may leads to serious scalability problems.
- Data Sparsity: Typically the number of users in the recommender system is much larger than the number of items. As a result, the user-item matrix can get very sparse. This poses serious algorithmic challenges for the performance of the recommender system.

Answer 4.90: Yes, we can use linear regression here, because the coefficients β appear as linear coefficients.

Answer 4.91: When the number of features exceeds the number of data points, there is no unique solution. Hence, the OLS cannot constrain the parameters of the model and the algorithm will fail. To mitigate this situation, we may want to only include those features that are likely to be good predictors of our output variable, via PCA or variable ranking.

Another approach is to include a regularization term in the regression, e.g., lasso (Eq. C.7) or ridge (Eq. C.8). The regularization term will provide some extra constraints on the model.

Answer 4.92: Yes, it is possible. The intercept term shows model prediction without any independent variable, i.e., the mean prediction (labeled as "prediction" in Eq. C.9).

When intercept term is present, R^2 value evaluates the model with respect to this mean model. In the absence of intercept term, the denominator in Eq. C.9 can get smaller under certain conditions, resulting in a higher R^2.

Answer 4.93: It can in some cases, however, there are two caveats:

1. It needs to be done very carefully, e.g., converting city names into a continuous variable is very context dependent.
2. By treating discrete values as representation of a continuous distribution, we are adding extra information to the problem, which may be justified sometimes (e.g., student grades on a scale $0 - 100$).

I personally prefer to train separate regression model for each category if the number of categories is small, then combine them in some fashion.

Answer 4.94: Having low bias and high variance means that the model is too complex — with too many degrees of freedom — for the given amount of training data. When this model is tested on an unseen dataset, it is likely to give disappointing results. Therefore, we should use an algorithm that is simpler and has less number of free parameters.

This can be achieved using the following techniques:

- Reducing the number of features. This eliminates the most noisy features and makes the model simpler.
- Regularization, where higher model coefficients get penalized, hence lowering the model complexity.

- Using an ensemble learning algorithm, such as random forest, which generates a large set of models from repeated randomized sampling of data. Later, the model predictions can be combined using voting (classification) or averaging (regression), thus reducing high variance.

Answer 4.95: The choice of machine learning algorithm generally depends on the type of data under study. If the data exhibit linearity, and we need to predict the value of the dependent variable, then linear regression would be an appropriate algorithm to use. On the other hand, if it is a classification problem where features are mostly uncorrelated, then Naïve Bayes would be a natural choice. If the data exhibit nonlinearity, then a more complex algorithm, such as random forest, would be necessary. For data consisting of images, neural network based models may be more effective.

In many situations, which algorithm to use is not all that obvious. Then we may have to decide on the basis of cross-validation performance. Let's say we try four algorithms: Naïve Bayes, AdaBoost, neural network, and random forest. We split our data into 5 nonoverlapping subsets. Then, we iteratively train a model using 4 of these subsets and predict performance on the 5^{th}. Finally, we estimate the mean performance and variance of all subsets for each algorithm. The best performing one wins!

Answer 4.96: Again, we would use cross validation, as described in the answer to question 4.95, to prove the claimed improvement.

Answer 4.97: I am very likely using a constant, too slow learning rate. I can fix this by using an adaptive step size for gradient descent, where the learning goes fast when away from minima, but slow when close to a minimum.

Answer 4.98: It is very likely that the model is overtrained. It has small training error but large variance, thus resulting in large validation error. Training error 0.00 means that the classifier has mimicked training data patterns to such an extent that it tries to fit even the

statistical noise in the data! When this classifier was run on the validation sample, it couldn't find those same patterns, and therefore made predictions with much larger error.

To ameliorate this situation, we need to simplify the feature selection and perform cross-validation to improve predictability of the model. In the specific case of random forest, we can try to reduce the number of trees, or trim the trees to have smaller depths.

Answer 4.99: Yes. Single decision tree often overfits the data, making it a poor predictive model. An ensemble of decision trees is a powerful predictor by averaging away the errors of the individual trees.

Answer 4.100: This is possible if the decision tree training was either suboptimal or it overfitted the data. Time series data is known to posses linearity. Decision tree algorithm can fit both linear and nonlinear data. However, in this particular case the decision tree algorithm couldn't map the linear relationship as good as the regression model did. Linear regression is certainly capable of providing robust prediction if the data set satisfies its linearity assumptions.

Answer 4.101: Loading the data should not be a problem. The memory needed for 20 million keywords of 8 bits each and 10 million floats of 32 bits each is about 60 MB. Clustering this much data is a more challenging task. Clearly, we can't do any clustering with worse than N^2 complexity. Perhaps an approximate nearest neighbor or locality sensitive hashing is our best bet.

Appendix E

Natural Language Processing Answers

This appendix contains answers to the questions posed in Chapter 5.

Answer 5.1: Natural language processing is a field of artificial intelligence, concerned with the interactions between computers and human languages. Modern NLP algorithms are based on machine learning. So, instead of using a hard-coded set of rules for language-processing tasks, the NLP aims at statistical inference through the analysis of large corpus of words and documents [49, 50].

Answer 5.2: Basically, we first look for patterns in the unstructured data. Then group them into categories. Build search route patterns using graph theory and finally use this to extract structure from a new text data [51, 52].

Two well known statistical models to accomplish these steps are

1. Latent Dirichlet Allocation
2. Latent Semantic Indexing

Answer 5.3: It depends on the application needs. Direct access to text file can be fast and efficient if all we need is several inserts, sequential reads, and no concurrency. On the other hand, if we need concurrency, non-sequential read/write, atomicity, etc., then we are better off with a database.

Answer 5.4: Following are some of the models to characterize a body of text:

- Bag-of-words
- Vector Space Model (VSM)
- Term Frequency — Inverse Document Frequency (TF-IDF)

Answer 5.5: *Bag-of-words* is a simple representation, in which the text is represented as a bag of its words, disregarding grammar and word order but keeping multiplicity.

Answer 5.6: Given the bag of words, one can create feature vector for the text, where each feature is a word with the corresponding frequency — or any other metric — as the weight. Such a representation is called *Vector Space Model*.

	Bag of words	Vector space
Representation	Unordered set	Ordered terms
Term weights	Raw frequency	TF-IDF
Query method	Search terms	Vector algebra
Similarity metric	None	Semantic distance
Term importance	No ranking	Natural ranking

Table E.1: Bag of words vs. vector space model.

Answer 5.7: *Term frequency — inverse document frequency* is a special form of VSM representation for a document, where the features are the words weighted by their TF-IDF score. It is a numerical statistic to evaluate how important a word is to the document. It is often used as a weighting factor in information retrieval and text mining [53].

$$\text{TF-IDF}(\text{term}, \text{doc}, \text{corpus}) = f(\text{term}, \text{doc}) \times \text{IDF}(\text{term}, \text{corpus}), \tag{E.1}$$

where the term frequency, $f(\text{term}, \text{doc})$, is the raw frequency of the given term in the document. The inverse document frequency, IDF, is the logarithmically scaled inverse fraction of the documents that contain the given term.

Answer 5.8: The TF-IDF score is computed using the following formula

$$TF\text{-}IDF = f_{t,d} \times \log\left(\frac{N}{1 + |d \in D : t \in d|}\right), \tag{E.2}$$

where

$$\begin{aligned} f_{t,d} &= \text{Raw frequency of term } t \text{ in the document } d \\ N &= \text{Total number of documents in the corpus } N = |D| \\ |d \in D : t \in d| &= \text{Number of documents where the term } t \text{ appears} \end{aligned}$$

Answer 5.9: *word2vec* is a two-layer neural network that processes text. Its input is a text corpus and its output is a set of vectors — feature vectors for words in that corpus. Specifically, *word2vec* models are trained in a discriminative manner, such that probability of a word depends upon the neighboring words. This is trained by maximizing the log likelihood of actual correct contexts vs. a random context.

Answer 5.10: An n-gram is a contiguous sequence of n items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words, or base pairs. The n-grams typically are collected from a text or speech corpus.

An n-gram can be used to model sequences in NLP using the statistical properties of n-grams. Such models can answer questions like — *given a sequence of letters, what is the likelihood of the next letter*. Thus, an n-gram model is a type of probabilistic language model for predicting the next item in an $(n-1)$-order Markov chain.

Answer 5.11: *Latent Dirichlet Allocation* is a topic model that generates topics based on word frequency in a document. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. Thus, LDA is a generative probabilistic model, that assumes a Dirichlet prior over the latent topics.

Answer 5.12: Short documents have only a few words. Since LDA models the document as a mixture of topics based on word frequency, there are too few observations to infer the parameters of the model.

Answer 5.13: *Latent Semantic Indexing* is an indexing and retrieval method used to determine the relationship between terms and concepts contained in a document. LSI assumes that words that are close in meaning will occur in similar pieces of text. The matrix containing word counts per paragraph (rows: topic word, columns: paragraph) can be reduced, using truncated SVD, to contain only a small number of relevant terms. LSI is, thus, a method for dimensionality reduction to learn latent topics in a document.

Answer 5.14: Both LSI and LDA give a mechanism for determining topics from a corpus, and expressing documents in terms of those topics. However, LDA is probabilistic, so returns different topics with different probability distributions over the dictionary words. LSI being deterministic, gives the same topics and same distributions after every iteration.

In practice, LSI is much faster to train than LDA, but has lower accuracy. LSI also has the nice property that the topics are nested — once you compute N topics, you get a model for $\frac{N}{2}$ topics for free, with no extra computation. This is not true for LDA, where you'll need to retrain when changing the dimensionality.

Answer 5.15: A simple model would consist of the following steps:

1. Split the sentence into words, normalize them, and build a dictionary.
2. With each word, store how many times they occurred in tweets about the company, and how many times they appeared in tweets about the fruit. These tweets must be confirmed by a human.
3. When a new tweet comes in, find every word in the tweet in the dictionary and calculate their combined weighted score. Words that are used frequently in relation to the company would get a high company score, and vice versa.

Answer 5.16: We can use supervised learning. We would need a set of hand-tagged training data to learn features to distinguish between quotes and non-quotes. Some possible features could be

- First and last words in a sentence
- Does the sentence contain attribution words (*said, asked,* etc.)?
- Does it contain quotation marks?
- How many words does it have in quotes? Etc.

Answer 5.17: Let's assume that the error is due to misspelling of one or more words in the generated text. We can break down the text into word-tokens and submit each as a query to search engine (say, Google or Bing). A returned spelling suggestion would imply misspelled word that need to be replaced by the suggested correction. Otherwise, we will continue with the next token until all tokens get validated. Then, we output the full text with corrected spellings.

If the error is due to more subtle language features, then we would need to perform some kind of supervised learning to detect each kind of error and and come up with rules (or NLP models) to correct them.

Answer 5.18: This question is in the mold of an open-ended system design question. There can be multiple ways to answer this with emphasis on parts that the interviewer is more interested in.

In general, an automatic translation software system interprets the structure of sentences — i.e., arrangement of subject, verb, object, articles, prepositions etc. — in the source language and generates a translation based on the rules of the target language. The process involves multiple steps such as

- Breaking down complex and varying sentence structures
- Identifying parts of speech
- Resolving ambiguities
- Synthesizing the information in the structure of the new language

Each of these steps can be analyzed in further detail and would likely involve using a combination of rules, NLP models, or supervised learning.

Answer 5.19: Using LDA or SLI. These still return a set of relevant topic words. So, we either need a human interface to manually write the subject line based on these topic words, or encode a set of predefined rules, or even try training a machine learning model.

Answer 5.20: Stop words are words which are filtered out when processing natural language data. Although there is no single universal list of stop words, search engines typically filter out words like *to, I, has, the, be, or,* etc. Stop words are fundamental to information retrieval. Removing them from index increases both performance and the relevance of search results, because it results in fewer terms in the dictionary.

However, there are situations where we may not want to remove stop words, e.g., if we want to build a list of phrases and aim to support phrase search.

Answer 5.21: For this kind of sentiment analysis we use supervised learning to classify a text as positive or negative. We can start with a simple model — like Naïve Bayes — and counts of unigram from the bag of words as features. To make the model perform better, we could look at n-grams instead. We could remove punctuation and other non-characters. We could remove stop words. Perhaps we can improve further by using neural networks or an ensemble method like random forest to model our data.

Answer 5.22: In physics, entropy is a measure of the disorder in a system. In information theory, entropy is the uncertainty regarding which symbols to chose from a set of symbols, given their *a priori* probabilities. Since information is a decrease in uncertainty, we may regard entropy as the information required to construct the correct set of symbols. Mathematically,

$$H = -\sum_i p_i \log_2 p_i, \tag{E.3}$$

where p_i is the probability of state i.

Answer 5.23: We can use Eq. E.3 to compute n-gram entropies for English language. If spaces and punctuation are ignored, we have a twenty-six letter alphabet.

- For $n = 0$:

$$H_0 = \log_2 26 = 4.60 \text{ bits per letter}$$

- For $n = 1$:

$$H_1 = -\sum_{i=1}^{26} p_i \log_2 p_i = 4.14 \text{ bits per letter}$$

- Similarly, entropy for higher n-grams can be computed if we know their probabilities:

$$H_2 = 3.56 \text{ bits per letter}$$
$$H_3 = 3.3 \text{ bits per letter}$$
$$H_3 = 3.3 \text{ bits per letter}$$
etc.

Answer 5.24: PageRank [54] is a way of measuring the importance of website and is named after one of the founders of Google. The algorithm works by counting the number and quality of links to a page to determine how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

The PageRank algorithm assigns each web page p_i a numeric score, based on the following formula

$$\mathcal{P}(p_i) = \left(\frac{1-d}{N}\right) + d \sum_{p_j \in \text{pages linked to } p_i} \frac{\mathcal{P}(p_j)}{L(p_j)}, \qquad \text{(E.4)}$$

where $\mathcal{P}$ denotes the PageRank score, d is a damping factor usually set to 0.85, N is the total number of websites under consideration, and $L(p_j)$ = number of outbound links on page p_j.

Answer 5.25: Dependency parsing is a popular approach in NLP to analyze the grammatical structure of a sentence. A dependency parse

connects words according to their relationships. Each vertex in the tree represents a word. Child nodes are words that are dependent on the parent. Edges are labeled by the relationship.

Answer 5.26: Constituency parsing is a classical parsing, where words are leafs in the tree. Non-leaf nodes are constituents (e.g. noun-phrase, verb-phrase, etc.), but never words. The edges are unlabeled.

A constituency tree can be deterministically converted to dependency tree, if the head nodes are known. The other way conversion is not possible, since dependency trees have no information about which constituents should be created.

Answer 5.27: Deep parsing refers to building complete trees for a sentence, while shallow parsing refers to building a set of partial trees of one sentence each (e.g. grouping only noun phrases). Typically, dependency parsers produce complete trees, hence deep parsing, but it is possible for them to do shallow parsing.

Answer 5.28: Manual annotation used in these corpora is labor intensive, has large inter-annotator disagreements, and is hard to correct. These factors pose difficulties when mapping between the annotations and the original text. These difficulties can be mitigated by using a standard markup language (e.g., HTML, XML) for annotation and standard semantics for tagging.

Answer 5.29: Assuming the topics are known beforehand (e.g., Sports, Entertainment, History, and Politics), this is a classification problem. We can start with a simple Naïve Bayes classifier. We train the filter with some predefined corpus, and optionally provide a way for users to further refine it by flagging things that were incorrectly categorized.

We can start with the bag of words as features. If misclassification rate is too high, then we try improving feature selection — e.g., by removing stop words but keeping the phrases. To make the model perform better, we could try n-grams instead of the bag of words.

Answer 5.30: The primary advantage of unsupervised learning methods in NLP is that they do not require annotated data to learn a model. However, this advantage makes them difficult to evaluate against a manually labeled gold standard.

For this reason, unsupervised learning is well suited for basic tasks like document clustering and latent semantic indexing. For more substantial tasks, such as parts of speech tagging, named entry recognition, sentiment analysis, etc., supervised learning is almost a necessity.

Answer 5.31: The basic concepts and ideas of NLP would work for any language. However, much of the new advances in NLP are based on having lots of data, including annotated corpora. These are much more abundant for English than most other languages. Also, some languages like Arabic are heavily inflected. Therefore, before parsing an Arabic text one needs to do morphological analysis to generate word lemmas, which then provide texts similar to English. Lack of language specific parsers and reliable text corpora are the two main hurdles for doing NLP on a non-English text.

Answer 5.32: Although each language is unique, different languages often exhibit similar characteristics, e.g., phonetic, morphological, lexical, syntactic.

Answer 5.33: Which one is better depends on the task at hand. Character n-grams are popular because they are robust to spelling differences and need much less storage space. They are used in a wide variety of tasks, such as spelling correction, spam filtering, language identification, fuzzy string matching, etc. On the other hand, word n-grams are more suitable for analyzing occurrence of specific words in a larger chunk of text. They are used in searchable books corpus (e.g., Google Books), stop word removal, clustering, etc.

Answer 5.34: The process of assigning one of the parts of speech to the given word is called Parts Of Speech tagging (POS tagging). Parts of speech include nouns, verbs, adverbs, adjectives, pronouns, conjunction, and their sub-categories. POS tagger is a program that does this job.

Answer 5.35: POS tagging is a supervised learning problem, so classification algorithms (including SVM) should do as good as Markov models, or better. We are given a table of data, and we need to predict the values in the last column, i.e., *part of speech at word i*. The features can be derived using the Brown corpus, e.g., *part of speech at word* $i-1$, *last three letters of word at* $i+1$, etc. Then, we can use any supervised learning model of our choice.

Unknown words can be treated as a separate category, for which we would need external context to make prediction.

Answer 5.36: Some popular NLP packages in Python are *Scikit-learn* [33], *NLTK*, *Pattern*, *gensim*, *spaCy*, *polyglot*, etc.

Answer 5.37: We can start with some basic thresholds — such as, time-out every few seconds — and extract no more than the first 20 kB of each page. We can avoid revisiting pages already crawled. We can parallelize data normalization tasks and use NLP to explore semantic meanings and relationships.

Answer 5.38: Fuzzy merging is a process where we want to do a join on two tables A and B, but the key types may not be compatible or parts of data are missing in A or B. Usually, scripting languages like Python are better than SQL to handle this.

Answer 5.39: Because each text document contains a large number of words from the vocabulary, most document representations are extremely high-dimensional. In high-dimensional spaces, even the most basic clustering or similarity measures either fail completely or are too slow to be useful.

In addition, two similar documents often have very different word usages. This makes it difficult to directly use parametric algorithms, such as K-means. Our clustering algorithm also needs to be fast and easy to update in the case of streaming data. Latent Dirichlet Allocation is the right approach in this situation, because it provides an interpretable lower dimensional representation of documents.

Answer 5.40: There are two major flavors of *word2vec* — the skip-gram model and the continuous bag of words model. The goal of the skip-gram model is to provide vector representations of words that best predict a window of surrounding words. Therefore, it aims to maximize the probability of any context word, given the current center word.

Answer 5.41: Lexical ambiguity is the presence of two or more possible meanings of a single word. For example, treating the word *stand* as a noun or a verb.

Answer 5.42: Syntax level ambiguity is a situation where a sentence may be interpreted in more than one way due to ambiguous sentence structure. For example, the sentence "The professor said on Monday he would give an exam" has two potential meanings. It could be interpreted to mean that the professor would give an exam on the upcoming Monday. Or, it could mean that the professor said (on previous Monday) that he would give exam on some unspecified future date.

Answer 5.43: Preprocessing of text data generally includes the following steps:

- Eliminate extra white spaces, tabs, line-breaks, and non-printable characters.
- Convert text to lowercase to avoid distinguishing between words simply based on case.
- Remove punctuation, since it doesn't add value for bag-of-words based analysis.
- Remove numbers from the text, if they are not relevant to the analysis.
- Remove stop words.
- Perform stemming or lemmatization, based on the analysis context, to transforms each word in the text to its root word.

Answer 5.44: Fuzzy matching is the logic to match disparate phrases (or, terms, records in a database, etc.) with one another that either look

or sound alike, but are not spelled the same. For example, *"Barack Obama"* and *"Obama, Barack H."* are close enough to the human eye that they should be counted as similar. Fuzzy matching technique is used widely to make record-linkage, i.e., to compare two text records that refer to the same entity, but have less than 100% perfect match.

Answer 5.45: The three most commonly used stemming algorithms are: Porter, Snowball, and Lancaster.

Porter is the least aggressive of the three. It is also the oldest and most commonly used stemmer. Snowball stemmer is similar to Porter but with improved rules, faster, and slightly more aggressive. The stemmed representations are usually fairly intuitive, just like in the case of Porter. In both cases, "maximum" remains *maximum* and "saying" becomes *say*. Lancaster is the most aggressive and fastest of the three. The word stems can become so short that they are barely readable by a human anymore. For example, "maximum" becomes *maxim* although "saying" still reduces to *say*. This stemmer can be very useful in certain situations by hugely trimming down the working set of words.

Answer 5.46: Lemmatization refers to the morphological analysis of a word to remove its inflectional endings and return its basic dictionary form. For example, a lemmatizer should map "gone", "going", and "went" into *go*. In order to achieve this, the lemmatizer needs to know about the context of the word, i.e., whether the word is a noun, a verb, an adjective, etc. Therefore, typical lemmatizer implementations are based on a lexical knowledge base — like WordNet, which is a large lexical database of English language words.

Answer 5.47: String distance is a metric that measures distance (or, inverse similarity) between two texts for approximate matching or comparison. It provides a numeric score indicating the algorithm-specific indication of distance. The most widely used distance metric is Levenshtein distance, which is based on the edit distance between two strings. Other popular distance metric are Hamming, Jaro, and Jaro-Winkler.

Answer 5.48: The Levenshtein distance between two strings s and t is the number of deletions, insertions, or substitutions required to transform s into t. Mathematically, it is given by $\mathrm{d}_{s,t}(|s|,|t|)$, where

$$\mathrm{d}_{s,t}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ \min \begin{cases} \mathrm{d}_{s,t}(i-1,j)+1 \\ \mathrm{d}_{s,t}(i,j-1)+1 \\ \mathrm{d}_{s,t}(i-1,j-1)+1-\delta_{s_i,t_j} \end{cases} & \text{otherwise,} \end{cases} \tag{E.5}$$

is the distance between the first i characters of s and first j characters of t and $|s|$ and $|t|$ are the lengths of the two strings, respectively. The first element in the minimum corresponds to deletion, the second to insertion, and the third to substitution.

For example, the Levenshtein distance between *SATURDAY* and *SUNDAY* is 3, since we need at least three edits to change one into the other: delete *A*, delete *T*, and replace $R \to N$.

Although the Levenshtein distance defined above is an integer number, it can be normalized to give a similarity value using the formula

$$\mathrm{d}_{s,t}^{\text{normalized}} = 1 - \frac{\mathrm{d}_{s,t}}{\max(|s|,|t|)} \tag{E.6}$$

Answer 5.49: The Jaro distance between two strings s_1 and s_2 with m number of matching characters between them is given by

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise,} \end{cases} \tag{E.7}$$

where $|s_i|$ is the length of string s_i and t is half the number of transpositions, i.e., matching characters with different sequence order. Two characters from s_1 and s_2 respectively, are considered matching only if they are the same and not farther than $\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1$.

The Jaro distance is scaled between 0 (not similar at all) and 1 (exact match). For example, given two strings s_1 = *"MARTHA"* and s_2 =

"MARHTA", we find: $|s_1| = |s_2| = 6$, $m = 6$, $t = 1$. Therefore, $d_j = \frac{1}{3}\left(\frac{6}{6} + \frac{6}{6} + \frac{6-1}{6}\right) = 0.9444$.

Winkler modified this algorithm to support the idea that differences near the start of the string are more significant than differences near the end of the string. The Jaro-Winkler distance is defined as

$$d_w = d_j + \ell p(1 - d_j), \tag{E.8}$$

where d_j is the Jaro distance, ℓ is the length of common prefix at the start of the string up to a maximum of four characters, and p is a constant scaling factor often taken to be 0.1. Using the above example of *"MARTHA"* versus *"MARHTA"*, we have $d_j = 0.9444$, $\ell = 3$, $p = 0.1$. Therefore, $d_w = 0.9611$. The Jaro-Winkler distance metric is often used in record-linkage to compare first or last names in different sources.

Answer 5.50: The Levenshtein distance is often used in approximate string matching situations, where the goal is to find fuzzy matches for short strings in a longer text. These use cases include spell checkers, correction systems for optical character recognition, natural language translation, fuzzy string search in record-linkage, etc.

The Jaro-Winkler distance is designed and best suited for short strings such as person names, where the goal is to compare surnames and given names in the two strings. The metric can also be used for checking dictionary for valid words, for ZIP code verification in strings that contain dashes or extension digits, and to search for matching phone numbers in a phone directory.

Answer 5.51: The Hamming distance between two strings of equal length is the number of positions at which the corresponding characters are different. For example, the Hamming distance between *"MARTHA"* and *"MARHTA"* is 2, since there are two mismatches.

The Hamming distance is one of several string metrics for measuring the edit distance between two sequences. It measures the minimum number of errors that could have transformed one string into

the other. It is often used in telecommunication to count the number of flipped bits in a fixed-length binary word as an estimate of error. The Hamming distance is also used in bioinformatics as a measure of genetic distance between two DNA strings. In many situations, it is used to measure the similarity between two bit vectors of the same dimension.

Answer 5.52: The Jaccard similarity between two documents (or strings) s_1 and s_2 is defined as the size of the intersection of the two documents divided by the size of their union.

$$J(s_1, s_2) = \frac{|s_1 \cap s_2|}{|s_1 \cup s_2|} = \frac{|s_1 \cap s_2|}{|s_1| + |s_2| - |s_1 \cap s_2|} \tag{E.9}$$

If s_1 and s_2 are both empty, then $J(s_1, s_2) = 1$. Clearly, $0 \leq J(s_1, s_2) \leq 1$. The Jaccard distance is complementary to the Jaccard similarity

$$d_J(s_1, s_2) = 1 - J(s_1, s_2) = \frac{|s_1 \cup s_2| - |s_1 \cap s_2|}{|s_1 \cup s_2|} \tag{E.10}$$

We can use Jaccard similarity between two documents to detect plagiarism. To do this, we first tokenize each document by changing all letters to lowercase and removing stop words, duplicates, and punctuation marks.

Next, we generate a set of n-grams for each document, where $n = 2$ initially, but can be optimized to also include higher values. We then compute the Jaccard similarity score J for the two documents, using Eq. E.9, and compare it against the expected J distribution obtained from a large number of two randomly chosen set of documents. If likelihood of getting this score or higher is below a pre-determined threshold, say, 0.01%, then we conclude that one of the two documents has plagiarized material from the other.

Answer 5.53: It refers to ambiguity in the usage of pronouns, which in the context of a particular sentence, could mean two or more people or things. It is sometimes clear from the context which meaning is intended. For example, "Amanda went to Kylie. She said, *I am tired.*" It's not clear from the context exactly who is tired, Amanda or Kylie.

Answer 5.54: Semantic analysis is the process of drawing exact meaning, or the dictionary meaning, from a text. It is done by mapping syntactic structures and objects in the task domain with the aim to extract their language-independent meanings. The semantic analyzer disregards phrases such as *"hot ice cream"*.

Answer 5.55: Syntactic analysis is the process of analyzing a string of symbols or words in a manner that conforms to the rules of a formal grammar. For example, a sentence like *"The school goes to boy"* would be rejected by English syntactic analyzer.

Answer 5.56: The basic steps in data de-duplication are record-linkage and consistency checking. The process of comparing each N record in the database with every other record is (at least) quadratic in complexity, $\mathcal{O}(N^2)$. Additional activities, such as clustering and finding the minimum vertex, work to consolidate duplicate records, or to accumulate evidence of data errors, are also computationally hard. Given the complexity of these tasks, cleaning large-scale data sets is quite challenging, both computationally and in terms of human effort.

Appendix F
Distributed Systems & Big Data Answers

This appendix contains answers to the questions posed in Chapter 6.

Answer 6.1: The interviewer is interested in your experience of analyzing big data, possibly on a distributed system. Make sure to describe what problem you were trying to solve, the database, tools used for reading and analyzing data, and the final result.

Answer 6.2: Some tools for parallelizing machine learning algorithms are

- Use of GPUs
- Writing our own implementation using low level primitives/MPI
- Map-reduce [55]
- Apache Spark [43]
- Vectorization provided by Torch [45], TensorFlow [44], etc.

Answer 6.3: ACID stands for Atomicity, Consistency, Isolation, and Durability [56]. It is a set of properties of database transactions.

Answer 6.4: The CAP theorem states that it is impossible for a distributed computer system to simultaneously provide more than two

out of three of the following guarantees: consistency, availability, and partition tolerance [56, 57].

Answer 6.5: Locking and optimistic concurrency control are two commonly used methods to resolve concurrency control.

- In the case of locking, a transaction locks an object in the database before using it, so another transaction must wait.
- In the optimistic concurrency control, transactions can proceed without restrictions and check for conflicts happens just before the commit. There is no global writes. Whenever the first write to an object is requested, a copy is made, and all subsequent writes happen to that copy. After transaction is complete and validated, the locally written data are made global.

Answer 6.6: Here is one way to do this query

```
SELECT student,grade from table
    WHERE grade > SELECT AVG(grade) from table
```

Answer 6.7: Here is the query

```
SELECT e.name, m.name FROM Employee e, Employee m
    WHERE e.mgr_id = m.emp_id
```

Answer 6.8: Here is one way to do this query

```
SELECT MAX(Salary) from Employee
    WHERE Salary NOT IN (
        SELECT MAX(Salary) from Employee
    )
```

Answer 6.9: Here is one way to do this query

```
SELECT COUNT(*), sex from Employees
    WHERE DOB BETWEEN '01/01/1960' AND '31/12/1975'
GROUP BY sex
```

Answer 6.10: Here is one way to do this query

```
SELECT * FROM Employees a
    WHERE rowid = (
        SELECT MAX(rowid) FROM Employees b
            WHERE a.empno = b.empno
    )
```

Answer 6.11: Group functions are necessary to get summary statistics of a dataset. COUNT, MAX, MIN, AVG, SUM, and DISTINCT are all group functions.

Answer 6.12: These are different types of joins available in SQL:

- INNER JOIN returns all records that are common between both tables.
- LEFT JOIN returns all records from the left table, even if there are no matches in the right table.
- RIGHT JOIN returns all records from the right table, even if there are no matches in the left table.
- FULL JOIN returns all records when there is a match in one of the tables.

Answer 6.13: In SQL, the UNION operator can be used to combine the results of two or more SELECT statements. It removes any duplicate records from the result. Each SELECT statement within the UNION must have the same number of fields and similar data types.

The difference between UNION and UNION ALL is that UNION ALL doesn't eliminate duplicate rows. Instead, it just pulls all records from all tables fitting the query criteria and combines them into a table.

Answer 6.14: SQL, which stands for Structured Query Language, is a standard language for accessing and manipulating relational databases. MySQL is a relational database management system, like SQL Server, Oracle, Postgres, etc.

Answer 6.15: Generally SQL query results display duplicate values by default. The DISTINCT keyword can be used to return only distinct values.

Answer 6.16: It is a traditional database schema with a central table. Satellite tables map ID in the central table to lookup tables that contain information fields. Such schema is useful in real-time applications as it

saves a lot of memory. Sometimes star schema involves several layers of summarization to recover information faster.

Answer 6.17: An embarrassingly parallelzable algorithm is one where little or no effort is needed to divide the problem into a number of parallel tasks. Some examples include:

- Serving static files on a web server to multiple users at once
- Rendering of computer graphics, where each pixel can be handled independently
- Simulation of a large number of independent events
- Brute-force attempts to unlock a password, etc.

Answer 6.18: Some basic differences between relational database and HDFS are [58]

- Relational database (e.g., SQL) can handle only structured data, while HDFS can handle both structured and unstructured data [59].
- Generally HDFS can handle much bigger datasets than relational database.
- HDFS is deigned for automated queries and indexing, whereas SQL is better for command line analysis.

See Table F.1 for a quick SQL vs. NoSQL comparison.

	SQL	NoSQL
Type	Relational	Non-relational
Data	Structured tables	Key-value/JSON
Transaction	ACID	CAP theorem
Schema	Static	Dynamic
Flexibility	Rigid schema	Flexible
Scale	Vertical, down time	Horizontal, no outage

Table F.1: SQL vs. NoSQL comparison.

Answer 6.19: Here you can talk about a NoSQL database you are most familiar with, such as MongoDB, CouchDB, Dynamo, Cassandra, HBase, etc. [57]

Table F.2 provides a quick comparison between CouchDB and MongoDB.

	CouchDB	MongoDB
Data	JSON documents	BSON documents
Interface	HTTP/REST	Custom TCP/IP port
Query	Map-reduce, Views	Map-reduce
Concurrency	MVCC	Update in-place
Replication	Master-Master	Master-Slave
Written in	Erlang	C++

Table F.2: CouchDB vs. MongoDB.

Answer 6.20: RDD is the acronym for Resilient Distribution Datasets. It is a fault-tolerant collection of operational elements that run parallel. The partitioned data in RDD are immutable and distributed.

Answer 6.21: By asking this question, the interviewer is interested in your familiarity with common distributed computing tools. You can describe your use case and explain where these tools fit in the data science or machine learning pipeline.

Answer 6.22: Map-reduce is a framework to split the input dataset into independent chunks, which are processed by the map tasks in parallel. The framework sorts the outputs of the maps, which are then input to the reduce tasks.

Map-reduce is referred to as a "shared-nothing" architecture because separate nodes run their portions of the work on their own disks or partitions. Its main advantage is scalability. The main disadvantage is that each node has to accommodate the peak load for the data it owns.

Answer 6.23: Map-reduce works well in problems where we can analyze one subset of data at a time without dependence on other subsets, so that we can parallelize the task. For instance, to compute keywords frequencies in 1 terabyte of text data, we can divide into 1000 sets of 1

gigabyte each, analyze them in parallel, then aggregate to get the final result.

Answer 6.24: Map-reduce doesn't work well in problems where we need to analyze multiple subsets of data simultaneously. For instance, let's say we want to find correlation between stock prices of company X and 10 other companies, using a dataset consisting of 10^6 stock price listings of 10^4 stock symbols. We can't simply split the data into 10000 stock symbols and compute for each. This is a large class of problems that map-reduce wasn't designed to solve.

Answer 6.25: We can certainly do word count and TF-IDF in map-reduce, because we can perform these tasks on any subset of data at a time, so they are parallelizable. Since Naïve Bayes classifier ignores correlations between features, it is just a product of conditional probabilities. So, we can analyze one column of data at a time, in parallel. At the end we multiply the probabilities to get the final answer. Thus, Naïve Bayes can also be implemented using map-reduce.

Answer 6.26: A cluster is considered balanced if for each data node, the ratio of used space to the total capacity differs from the corresponding ratio for the cluster by no more than some threshold value. Load balance becomes especially important when handling large files of data or when hardware resources use is critical.

To insure good load balance, it is a common practice to move data blocks from a highly-utilized node to under-utilized node in an iterative fashion. In each iteration, a node moves or receives no more than the threshold fraction of its capacity and each iteration runs no more than, e.g., 20 minutes.

Answer 6.27: The framework splits the input data into independent chunks which are processed by the map tasks. The *Map*() function performs filtering and sorting on the input chunk of data. The *combiner* collects the output of the map tasks and calls the *partitioner* to write the collected output to disk, which is then shuffled and sorted before it can be reduced. The *Reduce*() function performs the aggregation operation on the output from the previous step.

Answer 6.28: *K*-means clustering is a popular algorithm that is easy to parallelize. In the standard (non-parallel) version of the algorithm, we are given *K* initial cluster centers and proceed in the following steps:

1. Compute Euclidean distance from each data point to the clusters and associate it with the closest cluster.
2. Recompute each cluster center to reflect the true mean of its constituents.
3. Repeat steps 1 and 2 until the convergence criteria are satisfied.

In the parallel version of the algorithm, since data live in multiple partitions on different nodes, each node is given *K* identical initial cluster centers. Then, we proceed in the following steps:

1. Perform standard *K*-means clustering on each partition.
2. Perform reduction for global centroids and take global mean square error as uncertainty.

Answer 6.29: We can use map-reduce for this. We split the dataset into 100 chunks, then sort each of them in parallel. Then, we do mergesort on the sorted arrays in the reduce step. Time complexity of this approach is $O(n \log n)$.

If we already know the range of the original dataset, then we can create a histogram of their counts and take the middle element as the median. This approach is $O(n)$, because we compute the histogram by going once through all the numbers and then find the median by going through the elements of the histogram.

Answer 6.30: If both forms of the solution are available, the closed form is generally preferable. However, sometimes the closed form involves large sparse matrices and their inverse, which can get computationally prohibitive. In this situation, using an iterative method is computationally more efficient.

Answer 6.31: Cloud security concerns relate to risk areas, such as external data storage, dependency on the public internet, and lack of

control. Cloud environments are also at a higher risk of data breach, hacked APIs, compromised credentials, and other such threats.

Answer 6.32: A vector is a one-dimensional array of elements. However, in many applications, the elements of a vector have mostly zero values. Such a vector is said to be sparse. It is inefficient to use a one-dimensional array to store a sparse vector. Often the sparse vector is stored as two parallel arrays — one for indices and the other for values.

Answer 6.33: Spark provides good performance for running an application in parallel on the Hadoop [59] cluster — for both on disk and in memory computation. It also supports advanced analytic options, like graph algorithms, machine learning, streaming data, etc. In addition, Spark has built-in APIs in multiple languages, such as Scala and Python.

Answer 6.34: Spark has its own cluster management computation and mainly uses Hadoop for storage.

Answer 6.35: A typical workflow is as follows:

- Create some input RDDs from external data.
- Transform them to define new RDDs using transformations like *map*(), *filter*(), etc.
- Ask Spark to *persist*() any intermediate RDDs that will need to be reused.
- Launch actions such as *count*() and *first*() to kick off the parallel computation.

Answer 6.36: Spark typically consumes large number of system resources, hence does not scale well for compute intensive jobs . Its in-memory capability is too limited for cost efficient processing of big data. Also, Spark doesn't have its own file management system, and hence needs to be integrated with other cloud based data platforms or Apache Hadoop.

Answer 6.37: Data storage model in Apache Spark is based on RDDs. The RDDs help achieve fault tolerance through lineage. They always have the information on how to build from other datasets. If any partition of an RDD is lost due to failure, the lineage helps build only that particular lost partition.

Answer 6.38: The RDDs in Spark typically depend on one or more other RDDs. The representation of dependencies between RDDs is known as the lineage graph. Lineage graph is used to compute each RDD on demand, so that whenever a part of the persistent RDD is lost, the lost data can be recovered using the lineage graph information.

Answer 6.39: A partition is a smaller and logical division of data, similar to "split" in map-reduce. Partitioning is the process to derive logical units of data to speed up processing. Everything in Spark essentially works on a partitioned RDD.

Answer 6.40: While *persist*() allows the user to specify the storage level, *cache*() uses the default storage level.

Answer 6.41: Transformations are functions executed on demand, to produce a new RDD. Some examples of transformations include *map*(), *filter*(), and *reduceByKey*(). Actions are the results of RDD computations or transformations. After an action is performed, the data from RDD moves back to the local machine. Some examples of actions include *reduce*(), *collect*(), *first*(), and *take*().

Answer 6.42: Transformations in Spark — e.g., *map*, *filter*, *groupByKey*, etc.— are not evaluated until we perform an action such as *reduce*, *collect*, *count*, etc. For example, when *map*() is called on an RDD, the operation is not performed immediately. This helps optimize the overall data processing workflow. Thus, Spark has a "Lazy Evaluation" philosophy.

Answer 6.43: No. When an RDD is created, it belongs to the Spark context it originated from and can't be shared between SparkContexts.

Answer 6.44: There are many ways to create RDDs from different input sources, e.g.,

1. Scala's collections: sc.parallelize(0 to 100)
2. Local or remote filesystems: sc.textFile("data.text")
3. Hadoop file system: using sc.newAPIHadoopFile

Answer 6.45: Spark runs jobs in parallel. The RDDs are split into partitions to be processed and written in parallel. Inside a partition, data is processed sequentially.

Answer 6.46: A DataFrame is an RDD composed of row objects with additional schema information of the types in each column. Row objects are just wrappers around arrays of basic types, such as integers and strings.

Answer 6.47: Spark relies on data locality, or proximity to data source. This makes Spark jobs sensitive to where the data is located. With HDFS, the Spark driver contacts *NameNode* about the *DataNodes* (ideally local) containing the various blocks of a file and their location. It then schedules the work to the compute nodes which are running on each storage element.

Answer 6.48: Some of the data sources Spark can process are:

- Hadoop File System (HDFS)
- Cassandra (NoSQL database)
- HBase (NoSQL database)
- S3 cloud storage

Answer 6.49: Spark Streaming helps to process live stream data. Data can be ingested from many sources like HDFS, Apache Kafka, Flume, S3, or TCP sockets, and can be processed using complex algorithms expressed with high-level functions, such as *map*(), *reduce*(), *join*(), etc.

Answer 6.50: Apache Kafka, Flume, Amazon Kinesis, etc.

Answer 6.51: YARN and Mesos are the two main clusters managers used by Spark. Mesos is preferable because of its rich resource scheduling capabilities. For example, when several users run interactive shells, Mesos scales down the CPU allocation between commands to optimize the resources.

Answer 6.52: There are three core components to Spark: Driver, Executor, and Cluster Manager.

- Driver runs the *main*() method of the program to create RDDs and perform transformations and actions on them.
- Executor runs the individual tasks of a Spark job.
- Cluster Manager is a pluggable component in Spark to launch Executors and Drivers. Mesos and YARN are the two most frequently used cluster managers.

Answer 6.53: Default level of parallelism is the number of partitions, when not specified explicitly by the user.

Answer 6.54: Spark can run only one concurrent task for every partition of an RDD. So, if we have a cluster with 100 cores, we want our RDDs to have at least 100 partitions.

Answer 6.55: Execution starts with the earliest RDDs, i.e., those with no dependencies on other RDDs or cached data. Execution ends with the RDD that produces the result of the action.

Answer 6.56: The maximum size of a partition is ultimately limited by the available memory of an executor.

Answer 6.57: Shuffling is a process of repartitioning data across partitions and may cause moving it across the Java virtual machines (JVMs). Shuffling should be avoided as much as possible, e.g., by using partial aggregation to reduce data transfer.

Answer 6.58: By using the *subtractByKey*() function.

Answer 6.59: Because *groupByKey*() shuffles all the data, which is slow. On the other hand, *reduceByKey*() shuffles only the results of sub-aggregations in each partition of the data.

Answer 6.60: When called on datasets of type (K, V) and (K, W), Join returns a dataset of $(K, (V, W))$ pairs with all pairs of elements for each key.

Answer 6.61: Some commonly used libraries based on the Spark ecosystem are

- Spark SQL (Shark)
- Spark Streaming
- GraphX
- MLlib, etc.

Answer 6.62: Here is one way to do this query

```
sc.textFile("data.txt").map(s => (s, 1)).reduceByKey(
    (a, b) => a + b).sortByKey().collect()
```

Answer 6.63: Here is one way to do this query

```
sc.textFile("hdfs://...").flatMap(line => line.split(" ")).
    map(word => (word, 1)).reduceByKey(_ + _).saveAsTextFile("hdfs://...")
```

Answer 6.64: Here is one way to do this query

```
val errors = sc.textFile("hdfs://...").toDF("line").
                filter(col("line").like("%ERROR%"))
//Count all errors
errors.count()

//Count the errors mentioning DivisionByZero
errors.filter(col("line").like("%DivisionByZero%")).count()

//Fetch the DivisionByZero errors as an array of strings
errors.filter(col("line").like("%DivisionByZero%")).collect()
```

Answer 6.65: Here is one way to do this query

```
//Create a DataFrame based on a table named "people"
val url = "jdbc:mysql://IP:Port/test?user=Username;password=Password"
```

```
val df = sc.read.format("jdbc").option("url", url).option("dbtable", "people").load
    ()

//Examine schema to make sure that the two needed columns are present
df.printSchema()

// Counts people by age
val countsByAge = df.groupBy("age").count()
countsByAge.show()

//Save countsByAge to S3 in the JSON format
countsByAge.write.format("json").save("s3://...")
```

Answer 6.66: We can generate a large number of random points in the rectangle with boundaries $((0,0),(a,0),(0,b),(a,b))$ and see how many fall inside the curve. This fraction (say, f) will be equal to the ratio of the areas of the curve to that of the rectangle. Hence, the area of the curve = $f \times a \times b$.

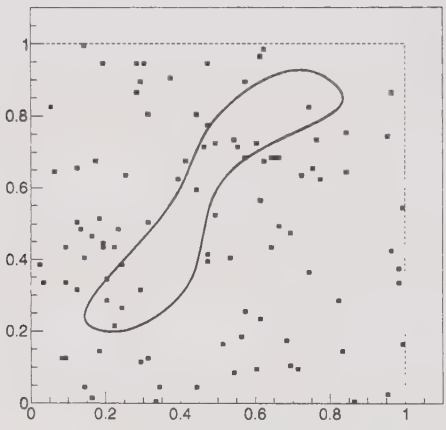


Figure F.1: Computation of the area of an arbitrary-shaped curve.

See Figure F.1 for an illustration of this technique. We want to compute the area of the shown curve, which is bounded in range $[0,1]$ on both axes. So, we choose a rectangle with $a = b = 1$ and generate a large number of random points. About 14% of the points fall inside the curve. Hence, the area of the shown curve is approximately $0.14 \times 1.0 \times 1.0 = 0.14$.

Answer 6.67: We can estimate the value of π by generating a large

number of random points in the unit square ($(0,0)$ to $(1,1)$) and see how many fall in the unit circle. The fraction should be $\pi/4$, so the value of π would be equal to four times the value of the fraction. Here is one way to compute this in Spark

```
val count = sc.parallelize(1 to 100000).map{i =>
    val x = Math.random()
    val y = Math.random()
    if (x*x + y*y < 1) 1 else 0
}.reduce(_ + _)

println("Pi is roughly " + 4.0 * count/100000)
```

Answer 6.68: Graph database is a database that uses graph structures for semantic queries with nodes, edges, and properties to represent and store data. Relationships are first-class citizens of the graph data model — unlike other database management systems, which require us to infer connections between entities using special properties, such as foreign keys (in the case of SQL) or out-of-band processing like map-reduce (in NoSQL).

Some commonly used graph databases are: Neo4j, ArangoDB, OrientDB, etc.

Appendix G

Culture-fit Answers (Selected)

This appendix contains answers to those questions in Chapter 7 that are not strictly personal preferences.

Answer 7.2: As someone who turns data into information, a good data scientist has a hacker's spirit. She has both technical acumen and statistical know-how. She can write code and work with teams to produce tools, pipelines, APIs, features, dashboards, and more.

She can sift through unfamiliar formats and legacy code, and develop new tools for the task if needed. A good data scientist can model data and make statistically-informed predictions and recommendations. But above all, a good data scientist is a creative problem solver.

Answer 7.12: Here are a few examples of best practices in data science:

- Before doing detailed analysis, make sure that we have the right kind of data and large enough data set for the desired accuracy.
- Choose the best tool for the job at hand! Rarely does a single tool handle all use cases well.
- Separate experiment from analysis and analysis from communication. It helps to consciously separate these three jobs.

- Use standard libraries for computation and machine learning as much as possible. This reduces programming error and has the advantage of large user-base to watch out for quality and correctness.
- Aim for reproducibility. Starting from the same raw data, we should be able to reproduce the analysis and obtain the same results.
- When sharing data, use a standard format with good coding-decoding support, such as JSON.
- Always use version control (e.g., git) to manage code.

Answer 7.13: A typical data science project involves the following steps

- **Plan the experiment:** This involves answering questions like "What's the objective of the project", "What kind of data", "What is the desired accuracy", ... etc.
- **Gather raw data:** Make sure that we have the right kind of data and large enough data set for the desired accuracy.
- **Gather tools:** For tasks such as data processing, cleaning, machine learning, statistical inference, plotting, summary, etc.
- **Execute experiment:** This step may involve running A/B tests, simulation, getting labeled or reference datasets, etc.
- **Analyze:** Use tools to analyze data.
- **Communicate:** This involves preparing summaries of experimental results, plots, publication, uploading processed data to repositories, etc.

Answer 7.38: You can be creative here! We all have subconsciously thought about problems where we wished we could get more data. Take health care, for example, where ML is being used for tasks like recognizing tumors in X-ray scans, but where digital data can be sparse. If there were no limitations, we could acquire data on every human being on the planet.

Similarly, if we could get a correctly tagged corpus of all the books

ever written in English language, then we would be able to solve so many of the NLP problems. If we could get unique digital footprints of all the devices in the world, then it would be so much easier to detect and stop cybersecurity threats. Sequencing the DNA of all life forms on earth will help immensely in solving some of the biggest problems in medicine and forensics.

Answer 7.51: Coding style is made up of many mundane decisions such as

- How and when to use comments
- Tabs or spaces for indentation
- Appropriate use of white space
- Proper naming of variables and functions
- Code grouping and organization
- Patterns to be used
- Patterns to be avoided

And many more. During the interview you can try to list some of these decision choices and indicate what you prefer.

Bibliography

[1] R. Sedgewick and K. Wayne. *Algorithms*. Addison-Wesley, 4th ed., 2011.

[2] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2009.

[3] S. Skiena. *The Algorithm Design Manual*. Springer, 2008.

[4] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes*. Cambridge University Press, 3rd ed., 2007.

[5] H. Warren. *Hacker's Delight*. Addison-Wesley, 2002.

[6] Khan Academy. *Introductory Computer Science Algorithms*. https://www.khanacademy.org/computing/computer-science/algorithms.

[7] S. Meyers. *Effective C++*. Addison-Wesley, 3rd ed., 2005.

[8] R. Martin. *Agile Software Development: Principles, Patterns, and Practices*. Pearson, 2002.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[10] K. Beck. *Test Driven Development: By Example*. Addison-Wesley, 2002.

[11] S. McConnell. *Code Complete*. Microsoft Press, 2nd ed., 2004.

[12] G. Meszaros. *xUnit Test Patterns: Refactoring Test Code*. Addison-Wesley, 2007.

[13] M. Fowler et al. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.

[14] M. DeGroot and M. Schervish. *Probability and Statistics*. Pearson, 4th ed., 2011.

[15] Wikipedia. *German tank problem*: https://en.wikipedia.org/wiki/German_tank_problem.

[16] Wikipedia. *p-value*: https://en.wikipedia.org/wiki/P-value.

[17] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[18] A. Gelman et al. *Bayesian Data Analysis*. Chapman and Hall, 3rd ed., 2013.

[19] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1996.

[20] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 3rd ed., 2009.

[21] E. Frank, M. Hall, and I. Witten. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 3rd ed., 2011.

[22] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[23] Y. Bengio, Y. LeCun, and G. Hinton. *Deep Learning*. Nature Vol. 521, 436, 2015.

[24] R. Callan. *The Essence of Neural Networks*. Prentice Hall, 1998.

[25] L. Breiman. *Random Forests*. Machine Learning Vol. 45, 5, 2001.

[26] Theano Development Team. *Theano: A Python framework for fast computation of mathematical expressions*. arXiv:1605.02688 [cs.SC], 2016.

[27] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[28] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 3rd ed., 1997.

[29] Andrew Ng. *Stanford CS 229 Machine Learning Course Materials.* http://cs229.stanford.edu/materials.html.

[30] R. Schapire and Y. Freund. *Boosting: Foundations and Algorithms.* MIT Press, 2012.

[31] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, 2nd ed., 2009.

[32] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 2nd ed., 1998.

[33] *scikit-learn: Machine Learning in Python.* http://scikit-learn.org.

[34] M. Kuhn and K. Johnson. *Applied Predictive Modeling.* Springer, 2013.

[35] Z. Zhou and J. Feng. *Deep Forest: Towards An Alternative to Deep Neural Networks.* arXiv:1702.08835 [cs.LG], 2017.

[36] K. Murphy. *Machine Learning: A Probabilistic Perspective.* MIT Press, 2012.

[37] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques.* MIT Press, 2009.

[38] I. Goodfellow et al. *Generative Adversarial Networks.* arXiv:1406.2661 [stat.ML], 2014.

[39] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction.* Bradford Books, 1998.

[40] *Weka 3: Data Mining Software in Java.* http://www.cs.waikato.ac.nz/ml/weka.

[41] *The R Project for Statistical Computing.* https://www.r-project.org.

[42] A. Hoecker et al. *TMVA - Toolkit for Multivariate Data Analysis.* arXiv:physics/ 0703039 [physics.data-an], 2007.

[43] S. Ryza, U. Laserson, S. Owen, and J. Wills. *Advanced Analytics with Spark.* O'Reilly Media, 2015.

[44] *TensorFlow: An open-source software library for Machine Intelligence.* https://www.tensorflow.org.

[45] *Torch: A scientific computing framework for LuaJIT.* http://torch.ch.

[46] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[47] A. Law. *Simulation Modeling and Analysis*. McGraw Hill, 4th ed., 2006.

[48] D. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[49] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[50] D. Jurafsky and J. Martin. *Speech and Language Processing*. Prentice Hall, 2nd ed., 2008.

[51] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[52] G. Ingersoll, T. Morton, and D. Farris. *Taming Text: How to Find, Organize, and Manipulate It*. Manning Publications, 2013.

[53] M. Russell. *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More*. O'Reilly Media, 2nd ed., 2013.

[54] L. Page. *PageRank: Bringing Order to the Web*. https://web.archive.org/web/20020506051802/www-diglib.stanford.edu/cgi-bin/WP/get/SIDL-WP-1997-0072, Stanford Digital Library Project, 1997.

[55] J. Dean and S. Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. https://research.google.com/archive/mapreduce.html, 2004.

[56] S. Faroult and P. Robson. *The Art of SQL*. O'Reilly Media, 2006.

[57] E. Redmond and J. Wilson. *Seven Databases in Seven Weeks*. Pragmatic Bookshelf, 2012.

[58] P. Sadalage and M. Fowler. *NoSQL Distilled*. Addison-Wesley, 2012.

[59] T. White. *Hadoop: The Definitive Guide*. O'Reilly Media, 4th ed., 2015.

Index

Made in the USA
Lexington, KY
10 April 2019

A collection of over *650 actual Data Scientist/Machine Learning Engineer job interview questions* along with their full answers, references, and useful tips

Kalanand Mishra, Ph.D., is a data scientist is San Francisco Bay Area. He has worked in both corporate and startup environments to bring AI-driven innovation in fields as diverse as healthcare, transportation, finance, and cybersecurity. He currently serves as Principal Data Scientist at a Fortune 500 company.

ISBN 9781727287325