A Practical Guide to

Artificial Intelligence and Data Analytics

Rayan S. Wali

Rayan Wali

A Practical Guide to Artificial Intelligence and Data Analytics

EDITION 1.0

ISBN: 979-87-9395-048-0

PUBLISHED BY RAYAN WALI

First printing, January 2022

Table of Contents

Foreword 6

About the Author 7

Acknowledgements 8

Part I: A Conceptual Illustration 10

- Practical AI and Data Analytics Introduction 11
- Fundamentals of Data Science 15
- The Data Pipeline 32
- The Machine Learning (ML) Pipeline 33
- Data Preprocessing 36
- Data Visualization 38
- Data Partitioning 43
- Python for Data Analysis 44
- Key Book Notations 50
- Probability and Statistics 51
- Statistical Data Analysis 81
- Calculus and Linear Algebra Basics 89
- Data Structures and Algorithms 94
- Machine Learning Models and Algorithms 116
- Deep Learning 140
- Evaluating Machine Learning Models 144
- Data Mining: Cluster and Outlier Analysis 150
- Classification and Regression in Python Usage 151

CONTENTS

Deep Learning for Computer Vision and NLP 158

Intro. to Reinforcement Learning 164

Time Series Data Analysis 165

A Deeper Dive into AI Systems 167

Databases and Cloud Computing 168

Reading Documentation Exercise 187

Applications of Data Analysis Exercises 189

Functional Programming for Data Analytics 191

Part II: Case Studies 199

Case Study I: Sports Web Scraping 200

Case Study II: Textual Analysis 214

Case Study III: Emergency Response Duration Analysis 216

Case Study IV: MNIST Image Classification 236

Case Study V: COVID-19 Statistical Data Analysis 242

Case Study VI: COVID-19 Chest X-Ray Screening 253

Case Study VII: Signal Strength Geospatial Analysis 279

Case Study VIII: NYC Crash Accidents Data Analysis 297

Case Study IX: Sales Forecasting 363

Case Study X: Meteorite Landings Analysis 374

Part III: Data Science and Analytics Skills Assessment 379

Exercises 380

Solutions 411

Foreword

Whether you are looking to prepare for AI/ML/Data Science job interviews or you are a beginner in the field of Data Science and AI, this book is designed for engineers and AI enthusiasts like you at all skill levels. Taking a different approach from a traditional textbook style of instruction, A PRACTICAL GUIDE TO AI AND DATA ANALYTICS touches on all of the fundamental topics you will need to understand deeper into machine learning and artificial intelligence research, literature, and practical applications with its three parts:

Part I: Concept Instruction
Part II: 10 Full-Length Case Studies
Part III: A Full-Length Assessment

With an illustrative approach to instruction, worked examples, and case studies, this easy-to-understand book simplifies many of the AI and Data Analytics key concepts, leading to an improvement of AI/ML system design skills.

I believe context is essential when introducing readers to more abstract content for the first time. Often, books do not place enough emphasis on practical applications and benchmarks for measuring learning. It is like providing readers directions on how to get to a place without showing them the landmarks that they need to reach their destination. At times, readers may become lost in the steps, when the instructions become more of a checklist rather than practicing critical thinking and problem-solving skills.

Integrating context, this book is designed to find the perfect balance between visualizations/illustrations and advanced mathematical and statistical concepts in delivering concepts.

I hope this book maximizes your learning experience and advances you in your AI journey!

About the Author

Rayan S. Wali is studying Computer Science and Applied Economics at Cornell University. He is an Educational Facilitator for Data Structures and Multivariable Calculus and a Teaching Assistant for Operating Systems. He has interest in Artificial Intelligence research and developing Machine Learning systems, and is constantly working on new projects and designing novel algorithms with the hope to improve current AI technologies.

He launched *A Practical Guide to AI and Data Analytics* in 2021, with the goal of improving the data science education. The ten practical case studies and the full-length assessment at the end of the book are designed to further strengthen data science and analytics concepts by giving diverse approaches to problems.

Alongside this book, he launched *Functionally Pragmatic Programming* in 2021. He will be joining Amazon Inc. as a Software Development Engineer in 2022.

For more details about the full-length assessment, please visit `https://rsw244.wixsite.com/dssa/`.

Acknowledgements

I would like to express my gratitude to Doug McKee, eminent Senior Professor at Cornell University, for inspiring me to write this book and steering me in the right direction with his valuable input. I gained a lot from the Active Learning educational style he implemented in the courses he taught at Cornell University. His teaching style and enthusiasm reignited my childhood passion of drafting word problems.

I would also like to acknowledge all my teachers at high school and professors at Cornell University for guiding me through the years.

Getting through my book required more than just academic support. I cannot begin to express my gratitude to my parents who had been unconditional in their emotional support and attention. I would also like to thank my mother who encouraged me to go outside the box and helped nurture my creativity.

Lastly, I would like to express my sincere thanks to one and all who directly or indirectly have lent their hands in this venture.

Part I: A Conceptual Illustration

A Practical Guide to Artificial Intelligence and Data Analytics

Rayan Wali
Cornell University

This book, titled *A Practical Guide to Artificial Intelligence and Data Analytics*, is designed for those who are interested in pursuing a data science career. It covers the most important topics of data science, from the data science workflow to probability and statistics and applications of machine learning, computer vision, and natural language processing.

1. What is Data Science?

Data Science, simply put, is the science of gathering and analyzing data. There are several methods of performing analysis once the data has been collected, using different tools and different strategies. We will go into some of the most important ones below.

2. Uses in the Real World

This book covers many of the concepts commonly practiced in the data science industry, with practical case studies and scenario-based exercises. Note that even though data science is practically performed on datasets, mathematical knowledge and programming basics are prerequisites to getting started on a full-length data science project. In addition, the information presented in this book will help you understand existing AI-themed research papers. And, a clear understanding of the AI literature will enable you to write your own AI research papers.

3. The Case Studies

Part II of this book consists of 10 full-length case studies. The purpose of these case studies are to deepen your understanding of AI in practice and to get you exposed to diverse solutions to a range of problems.

4. The Assessment

Part III of this book is the Data Science and Analytics Skills Assessment (DSSA). The purpose of this assessment is to ensure that you are

collecting user data, we would have been unable to improve newly-built frameworks with the power of machine learning.

7.2 Facial Recognition

Facial recognition is a common artificial intelligence application that strives to classify a given face, whether it is the person's emotions or their gender. Facial recognition softwares are used by law enforcement agencies to identify who is the suspect and who is the victim. At the same time, there are circulating privacy/ethical concerns, as sensors may collect data without the permission of those whose data is collected. And, in certain cases, this can potentially worsen in the case of misclassifications, or Type I/II errors (false positives/negatives). For example, a face classification algorithm could identify an innocent person as guilty, and this is partly a reason for why complete automation is not something that is agreed upon by everyone. Such privacy concerns are not only prevalent in facial recognition systems, but rather, are prevalent in other artificial intelligence applications, including the ones we will examine below.

7.3 Handwritten Text Recognition

Handwritten Text Recognition is part of Natural Language Processing (NLP), another sub-field of artificial intelligence. Given a stream of words of characters, it strives to extract useful information from it.

7.4 Robotics

Robotics plays a big part in our current semi-automated society. Artificial intelligence is embedded into robots, as they are able to learn from their mistakes. As they gather more data, they learn the surrounding environment, and thus, are able to perform the task they are given more accurately. Some common examples of robotics are self-driving cars, robotic vacuums, and medical imaging robots.

7.5 Financial Predictions

AI is commonly used to predict stock market prices. Several machine learning algorithms have been developed to extrapolate data and predict future trends and variability. AI is also used in quantitative trading by developing and applying algorithmic models to perform trades that make optimal decisions. An optimal decision might be one that is beneficial for the entire market. AI increases efficiency in financial predictions by not only increasing the speed of operations, but also increasing accuracy. Lastly, AI can also be used in detecting and preventing fraudulent transactions (similar to email spam filtering). The IBM Watson Studio is one of many solutions that predicts fraud in real-time with advanced deep learning technology.

9. The Data Pipeline

Below, we introduce the data pipeline — the fundamental steps to analyze data. The data pipeline can be applied to any kind of data analysis, whether it is pure statistical analysis or predictive analysis.

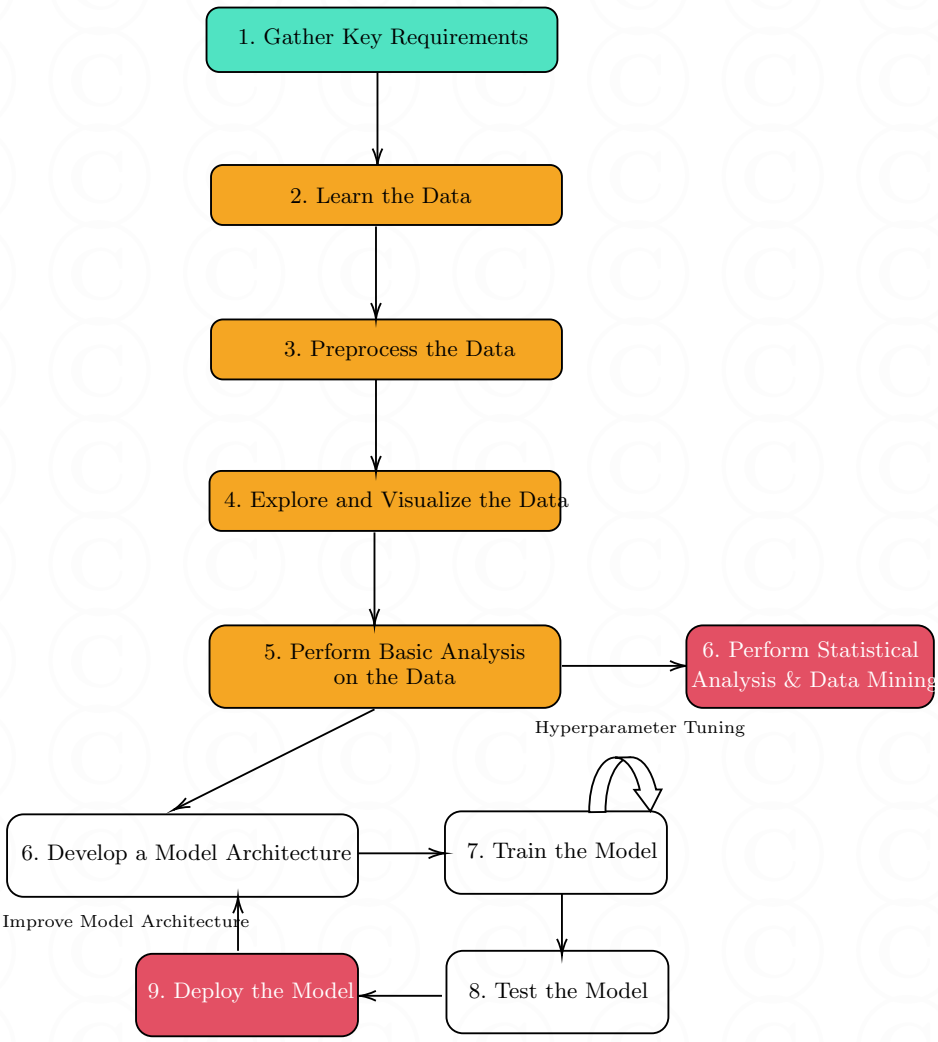


Figure 2
The Data Pipeline

10. The Machine Learning (ML) Pipeline

Below, we will go through the end-to-end ML pipeline, starting from the data collection phase to the model deployment phase. We will revisit and apply the ML pipeline to real-world situations in Part II (Case Studies) of this book. We will also define terminology that will be used in later parts of this book.

10.1 Terminology

Term	Definition
Instance/Example	An instance is one row of a dataset.
Features	A feauture is a property/field of a dataset.
Labels	A label belongs to a group of classes, and is assigned to an instance.
Epoch	An epoch is a single pass through all the instances of a dataset.
Model	An ML model represents a function that can take in an input and output a result.
Predictor Variable(s)	Predictor variable(s) is/are a model's independent variable.
Response Variable	A response variable is a model's dependent variable.
Predicted Labels	Predicted labels are labels generated by the model.
True Labels	True labels are labels assigned by people based on their observations.

10.2 Collecting & Cleaning Data

The first phase of the ML pipeline is data collection and cleaning. Data collection refers to gathering training and testing datasets that correspond to the problem statement you are examining. Once your data has been collected, you do not want to pass the raw data onto the training process. Instead, you want to clean it before. The techniques used for data cleaning really vary based on the anomalies present in the data collected, but there are certainly standard practices that are recommended to start with. This phase usually includes splitting the original training data into training and held-out validation sets. This phase may also include ensuring that the data is properly and consistently labeled.

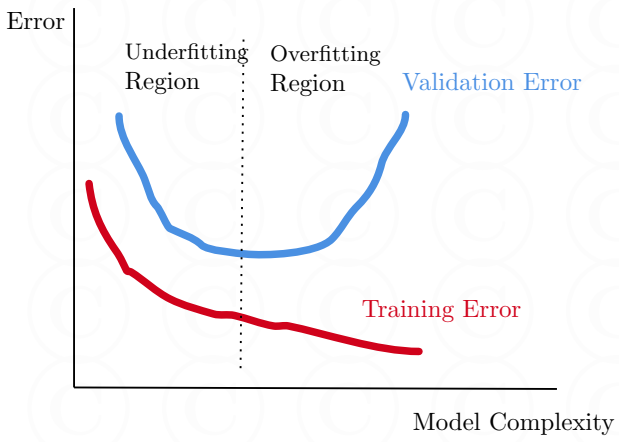
10.3 Training the Model

After further exploration of the data through visualizations and data analysis, the next phase of the ML pipeline is to produce the training dataset and train a chosen model on it. For now, you can think of a model as a function — which takes in an input and outputs a value. The model is constructed based on the type of ML problem you have, whether it is a classification, regression, or segmentation problem.

As more data and its corresponding labels enters your model, you hope that your model improves over the training process. This improvement of the training process is reflected in the training process by updating the parameters of the model.

All ML models are defined by two components: (1) a model architecture, and (2) a set of parameters. Training a model involves updating the parameters of a model. For a neural network/deep learning model architecture, the parameters are referred to as the weights of the model. The ML model architecture is based upon the model you select, and the set of model parameters is initially defined by you and is constantly updated as the training process continues. Other factors you can control which will affect the training process are known as hyperparameters. Examples of hyperparameters include the learning rate, the number of epochs the model is trained for, the objective loss function to specifically optimize your model on, and the optimizer type.

Now, we will define the notion of an error with respect to a set of predicted labels and a set of true labels. The training error is defined as the number of incorrect predictions divided by the total number of predictions, i.e., the proportion of incorrect predictions in the entire training dataset. Similarly, the validation error is defined the same way, except the evaluation is being conducted on the validation dataset instead of on the training dataset.



Sometimes, the training error is low and the validation error is high. This phenomenon is known as overfitting, and is a problem that ML

11. Data Preprocessing

As discussed in the preceding Data Pipeline section, once the data has been gathered, the data should be preprocessed before it is analyzed or inputted into a model. In this section, we will introduce the most common methods of preprocessing.

11.1 Handling Missing Values

You might encounter missing entries in the dataset you have collected, or even rows with missing fields. Missing values can be handled in several ways: filling in the missing values to dropping fields exceeding a certain proportion of missing values to removing rows that contain missing fields. The method which you choose should primarily depend on the construction of your dataset.

11.2 Field Normalization

Normalization is a statistical concept that is usually applied in the context of random variables. Random variables will be discussed in greater depth in a later section.

An example of feature normalization is feature scaling, which scales all the values of the column corresponding to the feature by a particular factor β. Unit scaling is when each element v_i of the column vector $\vec{v}$ corresponding to the field is recomputed as follows. Note that $||\vec{v}||$ computes the magnitude of vector $\vec{v}$.

$$v_i' = \frac{v_i}{||\vec{v}||}.$$

Another widely used form of feature normalization is min-max normalization, which scales each value v_i of a column vector $\vec{v}$ representing the feature to be scaled as follows:

$$v_i' = \frac{v_i - min(\vec{v})}{max(\vec{v}) - min(\vec{v})}.$$

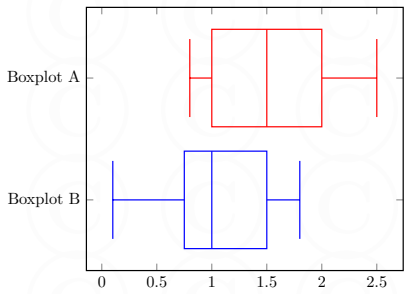
11.3 Ordinal Encoding

You may want to perform analysis that takes the correlation of two variables. What if one of the variable (field) is not an integer or a float? In this case, we can apply ordinal encoding to the field to retrieve a new field of integers. This process is described in detail as follows on a dataset storing a person's favorite movie:

Person ID	Favorite Movie
101	"Avengers"
102	"Star Wars"
103	"The Matrix"

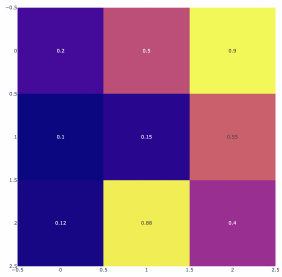
Ordinal Encoding →

Person ID	Favorite Movie
101	1
102	2
103	3



12.7 Heatmaps

Heatmaps describe pairwise correlations among random variables. Both axes of a heatmap include all random variables that are to be considered. A particular cell in the heatmap matrix with position (X,Y) denotes the correlation of random variables X and Y. The cell colors of a heatmap represent the correlation strengths. The correlation between any two random variables X and Y is denoted as $Corr(X,Y)$. When constructing a heatmap, you can set the correlations to either be a Pearson correlation or a Spearman correlation. The Pearson correlation measure is a better measure for testing a linear relationship among the variables, whereas the Spearman correlation is a better measure for testing a non-linear (polynomial, exponential) relationship among the variables. We will delve deeper into correlation and related statistical concepts in future sections of the book. The following is a heatmap with three variables:



The value within a cell of the heatmap above at row i and column j represents the correlation score between the random variable corresponding to row i and the random variable corresponding to column j. A lighter-colored cell of the heatmap represents a correlation score between two random variables that is high in magnitude, and a darker-colored cell of the heatmap represents a correlation score between two random variables that is low in magnitude.

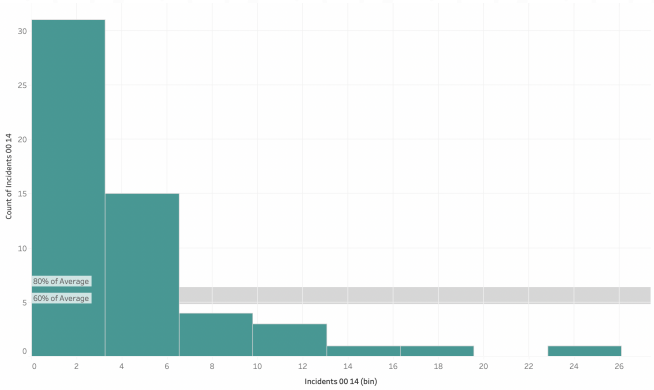
14. Data Visualization Exercises

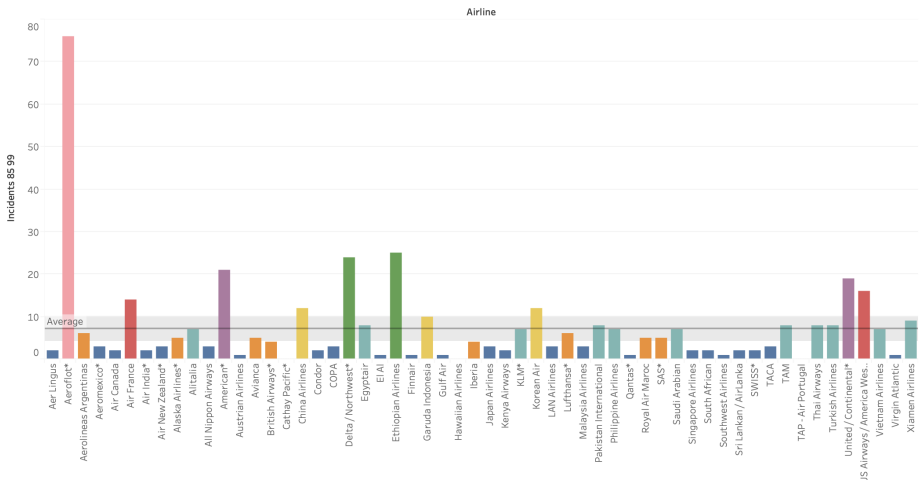
Consider a dataset containing airlines and their corresponding information including the total number of incidents and fatalities from 1985 to 1999 and from 2000 to 2014. The following are some visualizations produced on the dataset. Which of the following statements makes an incorrect claim regarding the visualizations?

A. The range of incidents between 1985 and 1999 is greater than that between 2000 and 2014.

B. The average number of incidents between 2000 and 2014 is lower than that between 1985 and 1999.

C. The distribution of incidents between 1985 and 1999 is skewed right.





Answer: (A)

15. Data Partitioning

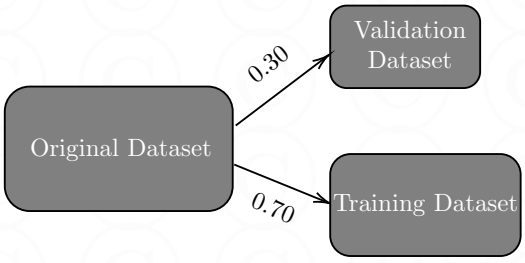
Data partitioning refers to splitting a dataset into exclusively reserved sub-datasets. In the context of machine learning, data partitioning refers to splitting a dataset into a training dataset and a held-out validation dataset. The training dataset is used to train your model, and the held-out validation dataset is used to evaluate the performance of your model. In the Data Pipeline in Figure 2, this process could take place anywhere between Steps 3 and 6.

A parameter that is often experimented with is the splitting ratio. The splitting ratio decides the exact percentage reserved for the training dataset and the exact percentage reserved for the validation dataset after the partitioning.

The splitting ratio is a hyperparameter that can largely affect the training and the validation accuracies, and hence, can decide whether overfitting or underfitting occurs. When deciding what splitting ratio to use, you should consider the composition of the dataset and determine approximately what sized portion of the original dataset might represent the entire dataset well.

In Python, data partitioning can be performed with the SkLearn library. The following is the Python code for splitting a dataset represented as a `DataFrame` with name `df` with a training dataset split size of 70%:

```
from sklearn.model_selection import train_test_split
train_data, test_data = train_test_split(df, train_size=0.70)
```



If you refer to the documentation for the `train_test_split` function, you will notice that the original dataset is randomly split into training and validation datasets. Note that in second line of the code above, `train_size = 0.70` could be replaced with `test_size = 0.30`.

16.1.3 Slicing NumPy Arrays. Slicing a NumPy array refers to selecting particular elements from the array. In NumPy, this is performed as follows:

```
arr[i:j] --- slices array [arr] starting from index [i]
and ending at index [j-1].
```

Below, we apply this to the array `arr` constructed above:

```
arr_sliced = arr[0:2] # selects the first two elements from
    [arr], storing the result in [arr_sliced]
```

16.1.4 Constructing a NumPy Array from Ranges. A sequence of numbers can be populated into a NumPy array with NumPy's `arange` command. The following describes its usage.

```
numpy.arange(a, b, c) --- creates a NumPy array
starting at value [a] (inclusive) and ending at value
[b] (non-inclusive) with a step size of [c].
```

```
arr_range = np.arange(1,5,2)
print(arr_range) # prints the NumPy array array([1,3])
```

16.2 The Pandas Library

The Pandas library in Python provides a `DataFrame` object that is commonly used to represent a dataset that is in `.csv` or `.xls` format. Throughout this section, we will be using the `df.csv` dataset, whose first three rows are shown below. The dataset stores people's name, age (in years), and height (in inches).

name	age	height
Larry	42	72
Alan	26	69
Kevin	22	65

16.2.1 DataFrames. The following is the code to convert a `.csv` file to a `DataFrame`.

```
pandas.read_csv('df.csv') --- reads [df.csv] as a
Pandas DataFrame object.
```

17. Python for Data Analysis Exercises

1. Consider the first four instances (rows) of the dataset `highway-work-orders.csv` containing information on highway work orders:

Highway	Miles_Complete	Total_Miles_of_Work
I-45	8	10
I-71	10	22
I-75	12	15
I-90	5	18

Write a function, `filter_highway`, in Python that returns a new Pandas `DataFrame` containing only highways with a completion rate of less than 50%.

Sample Solution:

```
import pandas as pd

def filter_highway():
    df = pd.read_csv('highway-work-orders.csv')
    return
        df[(df['Miles_Complete']/df['Total_Miles_of_Work']
        < 0.5)]
```

2. Consider the first three instances (rows) of the dataset `twitter-followers.csv` containing the top two followers of selected Twitter users. A user can *recognize* at most two users who are the user's top followers.

User_ID	Top_First_Follower	Top_Second_Follower
1	2	3
2	3	1
3	1	NaN

Write a function, `avg_user_recognitions`, in Python that returns the average number of users a Twitter user recognizes.

Sample Solution:

```
import pandas as pd

def avg_user_recognitions():
    df = pd.read_csv('twitter-followers.csv')
    return (df.count(axis=1) - 1).mean()
```

the covariance between two random variables A and B is denoted by $Cov(A,B)$.

19.2.7 Sample and Population Statistics. Sometimes, you might want to compute the expectation, variance, or another summary statistic of a random variable, but you cannot collect a complete set of data points for the random variable. The best you can do is to take a small sample of the population of data points, and use that sample to compute a sample statistic which will be used to estimate the population statistic you were originally looking for. A sample statistic is a measurement of a particular sample of a population. Just like a population of data points, a sample has its own mean, variance, etc. However, there is a slight difference in notation between sample and population statistics. The sample mean of a random variable X is denoted by $\bar{X}$, and the population mean of a random variable X is denoted by μ_X. The sample variance of a sample S is denoted by S^2 and the population variance of a random variable X is denoted by σ_X^2.

19.3 Distributions

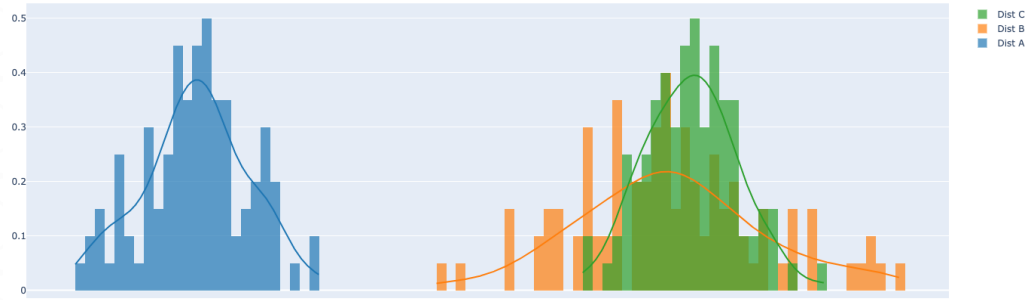
Statistical distributions represent a set of data that could be represented by a random variable. As discussed in Section 12: Data Visualization, a histogram represents a data distribution for discrete data.

Consider the following dataset on athlete ages in years:

Athlete ID	Age
1000	55
1050	31
1100	15
1150	18
1200	12
1250	22
1300	28
1350	11
1400	33
1450	42
1500	44
1550	49
1600	38

19.7 Statistical Distributions Guided Example

Consider the following three Gaussians distributions, each with 100 samples representing the time taken by a group of 100 competitive swimmers to swim 200 meters:



Before we begin our analysis, it is important to note that the raw data is naturally represented as a histogram, which is a collection of frequency bars. The curves that are drawn on top of the histogram, in this case, the blue, orange, and green curves, are smooth Gaussian curves fitted to the data.

19.7.1 Comparing Means. Given the distributions above, we can make the following observations regarding their means:

1. The blue-colored histogram represents a normal distribution with a lower mean than the orange and green colored histograms.
2. The green-colored histogram represents a normal distribution with a higher mean than the orange and blue colored histograms.

19.7.2 Comparing Variances. Given the distributions above, we can make the following observations regarding their variances:

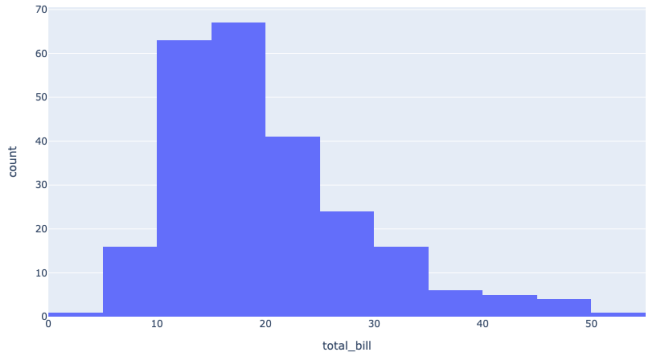
1. The green-colored histogram represents a normal distribution with a lower variance than the blue and orange colored histograms.
2. The orange-colored histogram represents a normal distribution with a higher variance than the blue and green colored histograms.

Since the standard deviation measure for a distribution is directly-proportional to the variance of a distribution, the relationships above also hold for comparing the standard deviations of the distributions.

19.7.3 Comparing Ranges. The range of the green-colored histogram is less than the range of the blue and orange colored

histograms, and the range of the orange colored histogram is greater than the range of the blue and green colored histograms.

19.7.4 Analyzing Skewness. Since a normal distribution does not have any skewness, i.e., no tail ranges more data values than the other tail, all three of the distributions above have zero skewness. The following is the distribution of the `total_bill` attribute of the built-in Seaborn `tips` dataset that is skewed right:



19.7.5 Data Generation Inference. Given the data above, we might want to recreate the process that generated the data distributions above. In order to do this, we should use the statistics we have compared across the distributions and ask why differences lie in particular statistics and not others.

For instance, we can observe that the mean of the blue distribution is significantly different from the means of the orange and the green distributions, whose means are close in value. In the context of our problem, we might associate this to the phenomenon of a difference in experience of a particular group of 100 swimmers, whose distribution of swimming times is represented by the blue-colored histogram. Similarly, with the variance, the orange-colored histogram has the most variability out of the three distributions, which might suggest that the group of 100 swimmers whose distribution of swimming times is represented by the orange-colored histogram might have a mix of experienced and less experienced swimmers.

However, note that swimmer experience level is one of many possible phenomenons for the difference in the distribution shapes.

The Poisson random variable holds a special property that both its expected value and variance are equal to the parameter λ.

Continuous Random Variables

The Exponential Random Variable
The exponential random variable denotes the time until the first success is obtained. The following is the probability density function (PDF) for an Exponential random variable:

$$f(X = x;\ \lambda) = \lambda e^{-\lambda x}.$$

The expected value of an exponential random variable is $\frac{1}{\lambda}$ and the variance of an exponential random variable is $\frac{1}{\lambda^2}$.

The Gamma Random Variable
A Gamma random variable models the time until the α^{th} success with rate λ, and is a special case of the exponential random variable. The shape of the distribution curves for a Gamma random variable cannot be easily determined as for other distributions, as the shape tends to vary significantly as the α and λ parameters change. Applications of the Gamma random variable include queuing theory, wait time modeling, and reliability analysis.

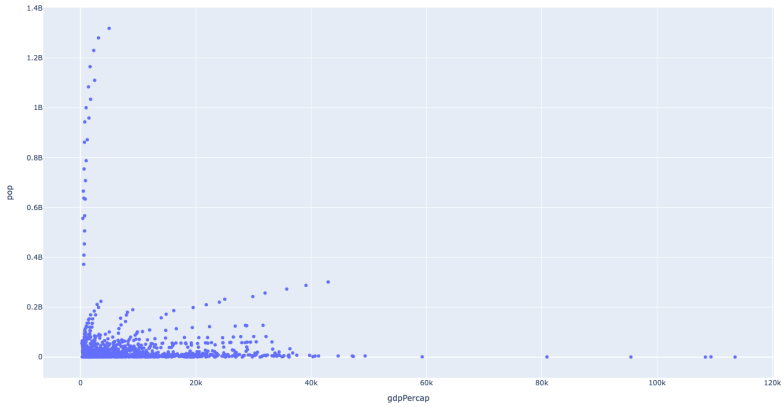
19.10 The Normal Distribution

Earlier in the book, we introduced the concept of a normal distribution and provided some examples. In this section, we will discuss some additional properties of normal distributions. Before defining standard normal distributions and the conversion process from a non-standard normal distribution to a standard normal distribution, we start by defining the notion of a z-score.

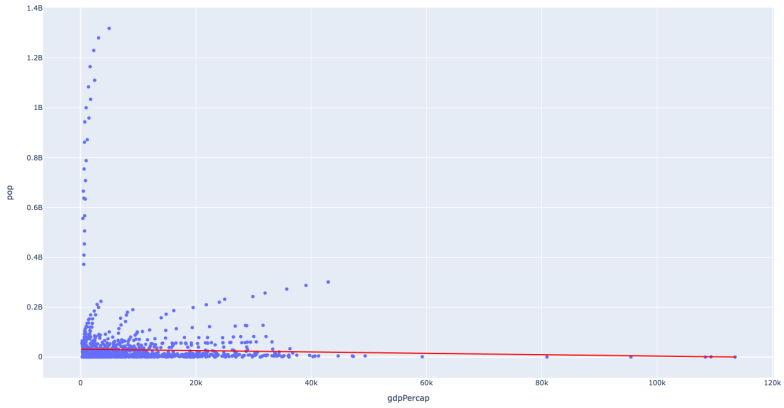
A z-score is a numerical measurement that is computed given a value from a normal distribution and a set of parameters of the normal distribution (mean and standard deviation), regardless of whether the normal distribution is standard or non-standard.

Guided Example. Consider a normal distribution representing the age of people in a town. The distribution has a mean of 50 and a standard deviation of 8. What is the z-score corresponding to a person from the town (and sampled by the normal distribution) with an age of 18?

19.13.1 A Practical Approach to Correlation and Covariance. Consider the following scatterplot plotting countries' per-capita GDPs and their populations.



We fit a linear trend curve through the data points in the scatterplot with the Ordinary Least Squares (OLS) Method (minimizes the sum of squared errors), and we obtain the following plot containing the original scatterplot along with a trend curve in red:



The trend curve above is downward-sloping and has a low magnitude slope. These properties of the trend curve indicate that there is a weak negative correlation between countries' per-capita GDPs and their populations.

In combinatorial optimization, the *likelihood function* is defined as $lik(\theta)=f(x_1,x_2,\ldots,x_n|\theta)$. This equation can be further simplified assuming the random variables $X_1,X_2,\ldots,X_n$ are i.i.d. as follows:

$$lik(\theta)=\prod_{i=1}^{n} f(x_i|\theta)$$

To find the most probable solution set, we would need to maximize the likelihood function above.

$$max(lik(\theta))=max\left(\prod_{i=1}^{n} f(x_i|\theta)\right)$$

Since maximizing this likelihood function can be quite lengthy, it is shown that maximizing the function above is equivalent to maximizing what is called the *log-likelihood function*. Knowing this, we set up a simplified maximization problem below. Solving the log-likelihood function below will yield the optimal solution(s) for the problem.

$$max(lik(\theta))=max\left(\sum_{i=1}^{n} log(f(x_i|\theta))\right)$$

Analysis. We will first compute the probability values for each of the 9 combination pairs in the form $\{X_1=x_1,X_2=x_2\}$, using the fact that the random variables are independently and identically distributed. We can thus define the joint density function as follows:

$$f_{X_1,X_2}(x_1,x_2)=\left(\theta(4-x_1)^{\theta+1}\right)\left(\theta(4-x_2)^{\theta+1}\right)$$

$\mathcal{P}(X_1=b_1,X_2=f_1)=0.272$, $\mathcal{P}(X_1=b_1,X_2=f_2)=0.171$,
$\mathcal{P}(X_1=b_1,X_2=f_3)=0.0773$, $\mathcal{P}(X_1=b_2,X_2=f_1)=0.171$,
$\mathcal{P}(X_1=b_2,X_2=f_2)=0.108$, $\mathcal{P}(X_1=b_2,X_2=f_3)=0.0485$,
$\mathcal{P}(X_1=b_3,X_2=f_1)=0.0773$, $\mathcal{P}(X_1=b_3,X_2=f_2)=0.0485$,
$\mathcal{P}(X_1=b_3,X_2=f_3)=0.0219$.

We can see that $\mathcal{P}(X_1=b_1,X_2=f_1)$ had the highest probability of 0.272. In other words, choosing the bar with brand b_1 and the flavor f_1 is the most probable event to occur. As per the problem statement, we want to find the probability that we do not get the most probable event. We will first find $\mathcal{P}(X_1\neq b_1 \text{ or } X_2\neq f_1)$. This is equal to $1-\mathcal{P}(X_1=b_1,X_2=f_1)$. Substituting the value we found above into this expression, we get $1-0.272=0.728$. Thus, since we want to find the probability that no one of the four bars are of brand b_1 and flavor f_1, we have $(0.728)^4=\boxed{0.281}$.

hypothesis, which is the negation of the null hypothesis."

Below, we will dive into more detail and will go through the steps needed to conduct a hypothesis test.

Step 1. Define the null hypothesis (H_0) and the alternate hypothesis (H_a).

Step 2. State the level of significance, α.

Step 3. Compute the test statistic.

Step 4. Using the test statistic, compute the p-value.

Step 5. Compare p to α. If $p \leq \alpha$, reject H_0; otherwise, fail to reject H_0. When $p \leq \alpha$, the result is referred to as statistically significant, since it suggests that there is strong evidence to reject H_0, the null hypothesis.

22.4 Bayesian Statistics

Bayesian statistics is a specific statistical approach based upon the notion of conditional probability. We will introduce conditional probability before we examine the Bayes' theorem.

As defined above, the probability of occurrence of event A given event B is denoted by $\mathcal{P}(A|B)$.

The Law of Total Probability

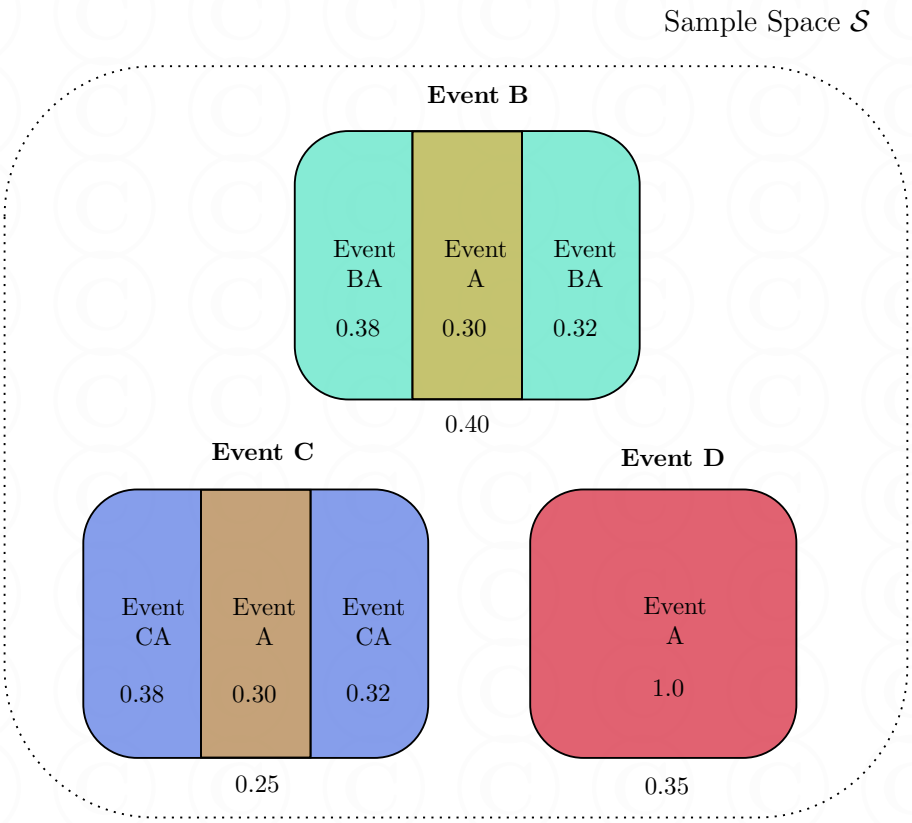
The probability of an event X, denoted by $\mathcal{P}(X)$ can be defined as follows for events $E_1 ... E_n$ such that $X \subseteq E_1 \cup E_2 \cup ... \cup E_n$: $\mathcal{P}(X) = \sum_{i=1}^{n} \mathcal{P}(X \cap E_i) = \sum_{i=1}^{n} \mathcal{P}(X|E_i)\mathcal{P}(E_i)$.

The Bayes' Theorem is the heart of Bayesian Statistics, and allows you to convert from one conditioned probability to another. The following states the Bayes' Theorem:

Bayes' Theorem

For events A and B, $\mathcal{P}(A|B) = \frac{\mathcal{P}(B|A)\mathcal{P}(A)}{\mathcal{P}(B)}$. Note that the Bayes' Theorem does not require independence between events.

Bayes' Theorem is used in the Naïve Bayes supervised machine learning algorithm. However, the Naïve Bayes algorithm adds a conditional independence assumption on top of the Bayes' Theorem, which we will discuss in detail in a later section.



In the figure above, there are three events in sample space $\mathcal{S}$: events B, C, and D. Each event contains sub-events, e.g., Event B contains Events A and BA. The probability of occurrence of Event A given the occurrence of Event B is equal to 0.30, and the probability of BA given event B is equal to $0.38+0.32=0.70$. Note that the following condition must hold for each outer event X and n sub-events within X.

$$\sum_{i=1}^{n} \mathcal{P}(E_i|X)=1.$$

Referring to the figure above and using the Law of Conditional Probability, what is the probability of occurrence of event A?

$$\begin{aligned}\mathcal{P}(A)&=\mathcal{P}(A|B)\mathcal{P}(B)+\mathcal{P}(A|C)\mathcal{P}(C)+\mathcal{P}(A|D)\mathcal{P}(D)\\&=(0.30\cdot 0.40)+(0.30\cdot 0.25)+(1.0\cdot 0.35)=\boxed{0.545}.\end{aligned}$$

23. Statistical Data Analysis Exercises

1. A package distribution facility receives packages in their inventory and process as many as possible in a particular interval of time. They receive four different batches of packages at different times, with each time having its own completion rate per package. Assume that the completion rate for any package is independent of that for any another package in the same batch. The following are probability mass functions produced for each of the four batch arrival times. It is known that the theoretical mean for each time equals 2. Which of the following statements make(s) a correct claim? Select all that apply:

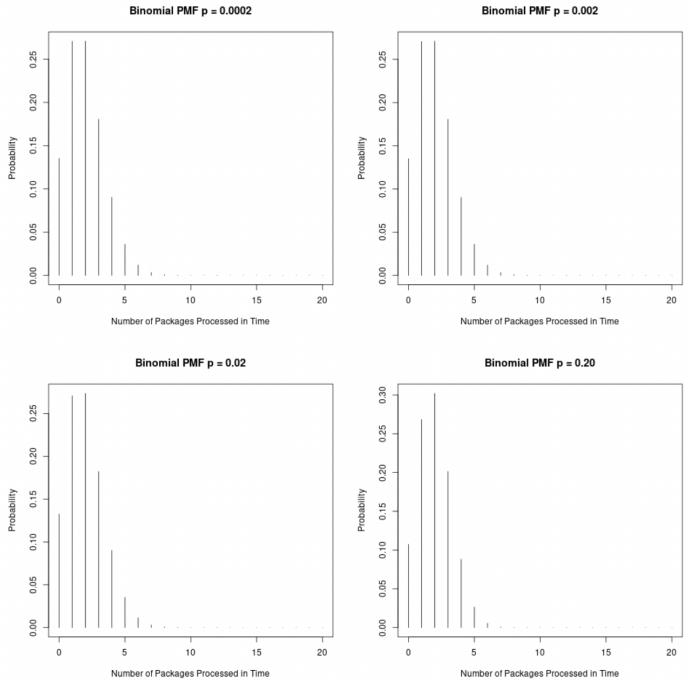


Figure 9
Binomial PMFs for the 4 batches

A. The mean for the distribution with $p = 0.0002$ is guaranteed to be the closest to the theoretical mean out of the four distributions since there are a large number of observations.

Component Analysis (PCA), which is a method of reducing a dataset's dimensionality. The principal components in PCA are vectors that are contained within a particular vector field.

In general, vectors can be used in any application that requires you to organize and structure data. Vectors are the building blocks of linear algebra, as a lot of linear algebra operations involve vectors, similar to how a lot of calculus applications involve integration. Vectors can also be useful to understanding the notion of tensors, which are essentially generalizations of vectors and are the building blocks of deep learning.

25.2 Matrices

Like vectors, matrices are mathematical objects that contain data. However, matrices, unlike vectors, may be composed of many rows and columns. A matrix with 2×3 dimensionality (2 rows and 3 columns) is defined as follows:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

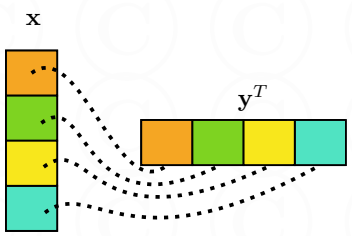
In data science, matrices are able to represent datasets, and are used to store data in a form that allows for performing future operations in a cheaper way by relying on matrix computation rather than on more expensive loops.

25.3 Dot Product

The dot product, also referred to as the scalar product, is an operation between two vectors. The dot product between two vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as follows, where $\mathbf{x}_i$ represents the i^{th} element of vector $\mathbf{x}$ and $\mathbf{y}_i$ represents the i^{th} element of vector $\mathbf{y}$:

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} \mathbf{x}_i * \mathbf{y}_i.$$

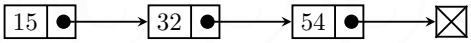
The following diagram illustrates the dot product between vector $\mathbf{x}$ and the transpose of vector $\mathbf{y}$, which is denoted by $\mathbf{y}^T$.



26.4 Linked Lists

Linked Lists are data structures that consist of a set of nodes and pointers from one node to another.

A singly linked list is a linked list with a single outgoing pointer for each node. A doubly linked list, on the other hand, is a linked list with two outgoing pointers for each node, one pointing to the next node in the linked list and one pointing to the previous node in the linked list. The following diagram is an illustration of a singly-linked list:

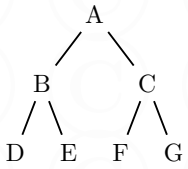


26.5 Trees

Tree data structures are characterized by a collection of nodes and edges between the nodes. All trees must satisfy the following invariant:

Invariant: The number of nodes of a tree is equal to the number of edges of the tree plus 1.

A binary tree is a tree where each node of the tree has at most two neighbors. The following diagram is an example of a binary tree with three levels, a root node A, and leaf nodes D, E, F, and G:



A binary search tree is a binary tree where the value of each node is greater than the value of the node's left neighbor and less than the value of the node's right neighbor. Binary search trees (BSTs) are often useful in optimizing the search process of a tree, e.g., looking up whether a node with a particular value exists in a tree. Insertions into BSTs are also optimized, as a node can be inserted into the tree by satisfying the BST invariant by traversing at most the height of a tree.

Trees are not typically traversed with loops, rather, they are traversed using recursion. The recursion traversal method will be described more in detail in the following section.

26.6 Graphs

Graphs are generalizations of trees. We defined trees as data structures that have no cycles. However, since graphs may contain cycles, all trees

In the table below, we can see that mergesort performed the best with respect to the worst-case running time out of the four sorting algorithms compared.

Table 2
Comparison of Sorting Algorithms

Mergesort	Selection sort	Insertion sort	Quicksort
$\mathcal{O}(nlogn)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$

Theorem 6. The Merge invariant states that n sets of elements can be merged into one sorted set if and only if each one of the n sets is already sorted.

The applications below are for you to reason through the approach you would take to solve the problems.

26.10 Application: International Trading

	Country ID	Current Value	Country 1	Country 2	Country 3
Country 1	C1	$32	true	false	true
Country 2	C2	$44	false	true	true
Country 3	C3	$25	true	false	true

Note: *The current values listed above are in billions.*

For a given country, find the smallest sequence of trades that yields that country the greatest benefit, i.e., the highest final amount. Print your answer to the console in a space separated format with the Country IDs the country should trade with, in order corresponding to the trading sequence. For the sequence of trades you found above, as a trading transaction between two countries is performed, update the current value column of the database for all countries. You may assume that countries other than the country of interest do not trade with each other, and that any two countries can trade with each other at most twice. The tabular data above is stored in a SQL server.

Trading between any two countries works in the following way: *if Country X trades with Country Y and Country X's current value is less than Country Y's current value, then Country X will gain half of the difference in the countries' values, and Country Y along with all other countries will lose an equal fraction of that amount.*

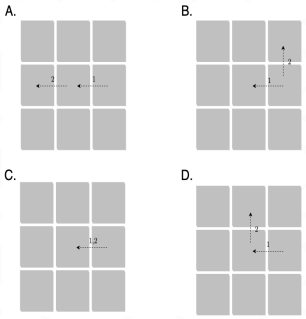
Exercises 6a and 6b refer to the following code in Java.

```
public class Cell {
    public Cell (int i, int j) {} // Instantiates a new Cell
        object at row i and column j, assuming that the
        origin is located at the top left of the grid.
    public int coins;
    public Cell left; // Moves to a new cell with location
        (i,j-1)
    public Cell right; // Moves to a new cell with location
        (i,j+1)
    public Cell up; // Moves to a new cell with location
        (i-1,j)
    public Cell down; // Moves to a new cell with location
        (i+1,j)
}

public int getNumCoins (Cell c) {
    int countCoins = 0;
    if (c == null) {
        return 0;
    }
    countCoins += c.coins;
    getNumCoins (c.left); getNumCoins (c.right); getNumCoins
        (c.up); getNumCoins (c.down);
    return countCoins;
}
```

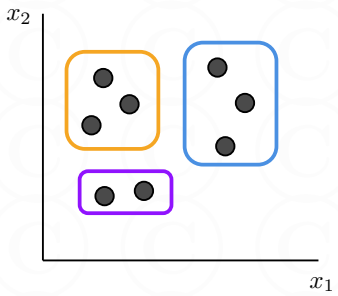
Exercise 6a. Does `getNumCoins` return the total number of collected coins in a grid as intended?
A. Yes B. No
Answer: *(B).*

Exercise 6b. Which of the following correctly maps the first two traversal steps of the algorithm on a 3×3 grid, with the starting location at (1,2)?



28.3 Clustering

Clustering aims to group similar data points into groups. They are naturally a form of unsupervised learning. Consider a dataset with the data points {`(1,2)`, `(1,1)`, `(2,-1)`}. Taking Euclidean distance as a similarity measure, and if we want a total of 2 clusters, the points `(1,2)` and `(1,1)` will be part of one group and the point `(2,-1)` will be part of the other group.



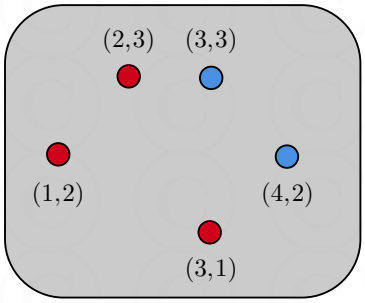
28.4 Segmentation

Segmentation decomposes an image into categories, separating out related objects. Essentially, it can be thought of as a clustering of similar objects. The first type of segmentation is instance segmentation. Instance segmentation considers different objects as different entities, regardless of whether the objects belong to the same class or not. For instance, in an image containing two cats, one dog, and a background of trees, the instance segmentation algorithm would identify four entities — the four objects. Another type of the segmentation, semantic segmentation, considers different objects belonging to the same class as different entities. Semantic segmentation applied to the image above would identify three entities — the two cats as one entity, the dog as one entity, and the background environment as one entity. The last type of segmentation, panoptic segmentation, is similar to instance segmentation, with a subtle difference that it does not allow entities to overlap.

30. Machine Learning Class Distributions

The concept of a class probability distribution is often used in machine learning, from the cross-entropy loss function to the Linear Discriminant Analysis (LDA) model. In this section, we will illustrate this concept using an example.

Consider the following dataset containing two-dimensional data belonging to two classes, red and blue:



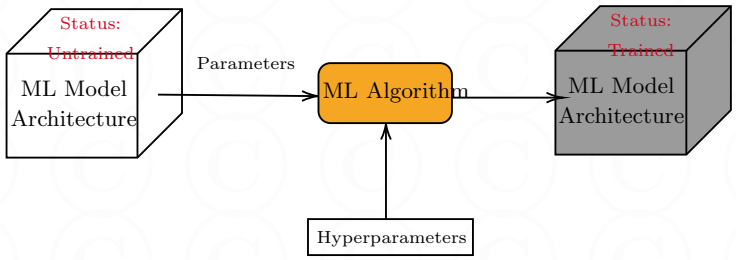
Recall from Section 18: Probability and Statistics that a distribution represents relative frequencies of data values. Below, we apply this concept to the dataset above by describing the distribution for both the red and blue classes.

We will start by analyzing the distribution of the red class. Since the data points are two-dimensional, we will generate a distribution for the x-values of points within the red class and a separate distribution for the y-values of points within the red class. The data points belonging to the red class are contained within the set {(1,2), (2,3), (3,1)}. The set of x-values is {1,2,3} and the set of y-values is also {1,2,3}. Since each value in both x and y distributions occur with a frequency of 1, we have an uniform distribution for both the x and y distributions for the red class.

We will now analyze the distribution of the blue class. Again, we will generate a distribution for the x-values of points within the blue class and a separate distribution for the y-values of points within the blue class. The data points belonging to the blue class are contained within the set {(3,3), (4,2)}. The set of x-values is {3,4} and the set of y-values is {2,3}. Since each value in both x and y distributions occur with a frequency of 1, just like for the red class, we have an uniform distribution for both the x and y distributions for the blue class.

32. Model and Algorithm Hyperparameters

Hyperparameters are additional parameters, which are not strictly considered parameters, that influence the training process of a machine learning model. The following diagram shows the complete training process starting from the development of a model architecture to having a trained model.



The parameters that result from the ML model architecture are inputted to the ML algorithm. The ML algorithm, alongside the model parameters, takes the hyperparameters as input. These hyperparameters include the number of epochs the model is trained for, the learning rate, the loss function for the model to be optimized on, and the type of optimizer. Each one of these attributes can take on a value, and that value influences how the model is trained. For instance, consider a neural network classification model classifying, given an image of a car, whether the car is of a particular model.

Now, consider the following initial state of hyperparameters:

- Learning rate $\alpha = 0.05$;
- 10 training epochs;
- SGD optimizer;
- 5 neural network layers;
- Binary cross-entropy loss function.

We now analyze the effect of changing these set of hyperparameters. If we increase the learning rate to $\alpha = 0.10$, the ML training algorithm might result in a sub-optimal solution. If we increase the number of epochs from 10 to 20, we might increase the training accuracy, but increase the risk of overfitting. If we change the optimizer type, the training loss curve over time would change. If we increase the number of neural network layers from 5 to 10, we could increase accuracy, but increase the risk of overfitting. Lastly, if we change the loss function, the trained neural network weights would change.

type of data you are given and what you would like to optimize.

Within linear regression, there are several optimization methods you can choose from, each with their own error functions. The most common ones are the least-squares method and the minimization of the sum of the absolute errors.

The least-squares method minimizes the following error function for a dataset with m data points:

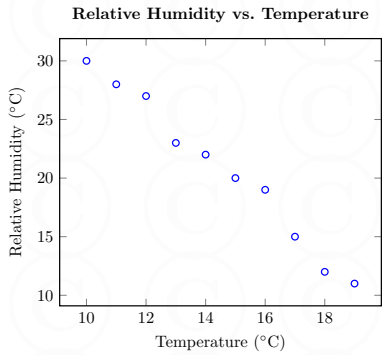
$$\min \sum_{i=1}^{m} \mathbb{E}(x_i, y_i) = \min \sum_{i=1}^{m} (f(x_i) - y_i)^2.$$

Similarly, the minimization of the sum of the absolute error is defined as follows for a dataset with m data points:

$$\min \sum_{i=1}^{m} \mathbb{E}(x_i, y_i) = \min \sum_{i=1}^{m} |f(x_i) - y_i|.$$

The choice of the error function above affects the fitted line that is predicted by linear regression, as the optimization objective now changes to another function.

33.1.1 A Practical Example. Consider the example introduced earlier of predicting the relative humidity level in degrees Celsius given the temperature in degrees Celsius. Our objective is to predict the relative humidity at a temperature, 5 degrees Celsius, that is not within the range of temperatures measured. In this case, we are extrapolating the relative humidity from the temperature.



classification problem classifying whether an individual is in the high-risk group for a certain infection.To traverse a decision tree, you must start from the root node and work your way to a leaf node of the tree (a node at the last level of the tree). Since the leaf nodes hold the classification result, you now have the classification output that the decision tree predicts. Note that to make a classification based on a decision tree, you always traverse exactly one root-to-leaf path of the tree. Additionally, if we combine two or more decision trees by assigning a weight to each decision tree to represent its say in the final prediction, we obtain a new ensemble model, the random forest.

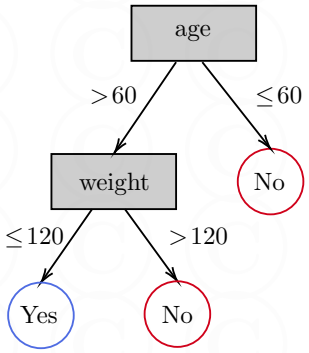


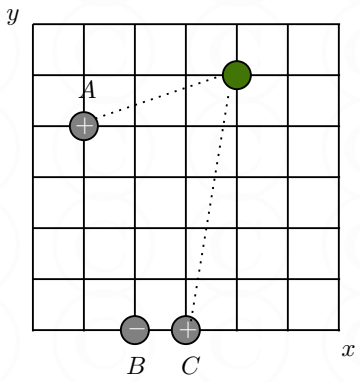
Figure 13
Sample Decision Tree

33.4 K-Means Clustering

K-Means Clustering is an unsupervised form of machine learning, since it does not require labeled data in order to be trained. The algorithm starts by selecting k random centroids and assigning data points to each cluster based on Euclidean distance. In the next iterations of the algorithm, the centroids are recomputed, and the process above is repeated until convergence, i.e., the centroid positions do not change. Given a set of data points (represented as tuples) $\{(x_1,y_1),...,(x_m,y_m)\}$ and k clusters, the k-means clustering algorithm goes through the following steps:

1. Randomly select k centroids.
2. Assign data points to each centroid based on minimum Euclidean distance.
3. Re-compute the k centroids with the updated mappings of data points.
4. Repeat steps 2 and 3 until the algorithm converges, i.e., the clusters do not change.

Consider the following example of k-nearest neighbors, with $k=2$ (2-nearest neighbors) and three points labeled A, B, and C and a test point located at coordinate (4,5):



The k-nearest neighbors prediction of the class label of the test point (4,5) will be the positive class, because both of its closest neighbors have a positive label, and hence, the majority is thus the positive label.

33.6 Linear Classifiers

A linear classifier produces a separating line, also known as a separating hyperplane, dividing the different classes. The data points on the separating hyperplane satisfy the following equality:

$$\vec{w}\cdot\vec{x_i}+b=0. \tag{5}$$

33.6.1 Classification Rule. The following equation states the linear classification rule:

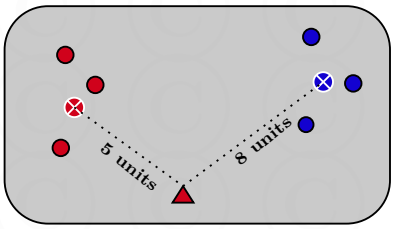
$$h_{\vec{w},b}(\vec{x_i})=\textbf{sign}(\vec{w}\cdot\vec{x_i}+b). \tag{6}$$

33.6.2 Radius and Margin Definitions.

- Radius R: the maximum length from the origin to any point in the dataset. $R=\max_i\ \vec{x}_i\cdot\vec{x}_i$.
- Margin γ: the smallest distance from any point to the hyperplane.
- The radius R and margin γ are used in placing upper bounds on the leave-one-out error for SVMs as well as in the Perceptron mistake bound.

33.6.3 Homogeneous Classifiers. A *homogeneous* classifier is one whose bias b equals 0.

predict given this labeled dataset and a particular test instance represented by the triangle in the diagram below.

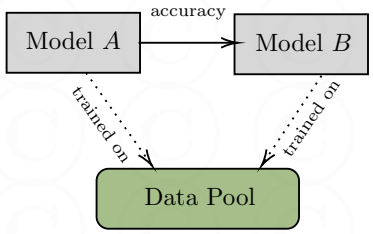


The LDA classifier will classify the test instance as part of the red class, since it was closer to the mean of the red class (5 units) than to the mean of the blue class (8 units).

33.10 Ensemble Models

Ensemble models are models that combine existing models to increase the level of generalizability. An example of an ensemble model that combines decision trees together is known as the random forest model. Ensemble models are categorized into three classes: bagging, boosting, and stacking. In this section, we will only discuss about bagging and boosting, since they tend to be the more popular methods.

33.10.1 Boosting. In boosting algorithms, different classification models, also referred to as learners, are trained, but in a sequential manner, where one model's accuracy influences the models that follow. Boosting reduces the variance of the predictions made by the model, and reduces the bias of the model in relation to a single learner.



33.10.2 Bagging. On the other hand, in bagging algorithms, a single model is trained independently on different views of a dataset to produce different models that are eventually merged together. Bagging

$$\mathcal{P}(1|X=110)=\frac{\mathcal{P}(X=110|1)\mathcal{P}(1)}{\mathcal{P}(X=110)}=\frac{\mathcal{P}(X=110|1)*1/3}{\mathcal{P}(X=110|0)\mathcal{P}(0)+\mathcal{P}(X=110|1)\mathcal{P}(1)}.$$

The Naïve Bayes assumption of *conditional independence* states the following:

$$\mathcal{P}(X=110|0)=\mathcal{P}(X_1=1|0)\mathcal{P}(X_2=1|0)\mathcal{P}(X_3=0|0), \text{ and}$$

$$\mathcal{P}(X=110|1)=\mathcal{P}(X_1=1|1)\mathcal{P}(X_2=1|1)\mathcal{P}(X_3=0|1).$$

$$\mathcal{P}(X=110|0)=(1/2)(1/2)(1/2)=1/8,\ \mathcal{P}(X=110|1)=(0)(0)(0)=0$$

$$\mathcal{P}(0|X=110)=\frac{1/8*2/3}{1/8*2/3}=1.$$

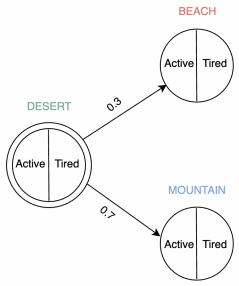
$$\mathcal{P}(1|X=110)=\frac{0*1/3}{\mathcal{P}(X=110|0)\mathcal{P}(0)}=0.$$

Since $\mathcal{P}(0|X=110)=1>\mathcal{P}(1|X=110)=0$, the Naïve Bayes Classifier will predict that a security breach will not occur.

33.12 The Hidden Markov Model (HMM)

The Hidden Markov Model (HMM) is a probabilistic form of the Markov model that solves sequence labeling problems. HMMs can be solved in a brute-force search way with graph traversals, but can also more efficiently be solved with dynamic programming algorithms such as the Viterbi Algorithm. Given an output sequence, the HMM finds a sequence of states to maximize the likelihood of occurrence for that specific sequence. The HMM is constructed by defining the *transition probabilities* and the *hidden state probabilities*, the former representing the probabilities of transitioning from one state to another, and the latter representing the probabilities of outputting a particular value in a given state. The optimization problem defined below consists of taking the products of the transition and the hidden state probabilities for the states visited, following the Multiplication Law of Probability.

33.12.1 HMM Example. You are living in a desert with your friend, who is planning a multi-leg trip. Your friend wants the first leg of the trip to either be to a mountainous location or one close to the beach. Where do you expect to go in order to maximize your chances of staying active throughout the first leg of your trip? The HMM and the hidden state probabilities representing your chances of activeness in each state are given in the following figure. *Note that the outgoing edges from the Beach and the Mountain states are hidden from the HMM below.*



	$\mathcal{P}$(Tired)
Desert	0.80
Beach	0.25
Mountain	0.65

Figure 15
HMM (left) and Hidden State Probabilities (right)

S^*, defined as follows, is the output of the HMM for a given output sequence $A_1,...,A_T$. We define events D, B, M for 'Desert', 'Beach', and 'Mountain' respectively, and A without a subscript as 'Active'.

$$S^* = argmax_{S\in\{S_1,...,S_T\}}\left(\mathcal{P}(A_1|S_1)\mathcal{P}(S_1)\prod_{t=2}^{T}\mathcal{P}(A_t|S_t)\mathcal{P}(S_t|S_{t-1})\right)$$

$$\mathcal{P}(AA|D\to B) = (0.20)(0.30)(1-0.25) = 0.045,\ \mathcal{P}(AA|D\to M) = (0.20)(0.70)(1-0.65) = 0.049$$

$$S^* = argmax_{S\in[\text{Beach, Mountain}],i\in[0,1]}\ (\mathcal{P}(AA|D\to S[i]),\mathcal{P}(AA|D\to S[1-i])) = \text{Mountain}$$

The results above indicate that you expect to go to the mountain for the first leg of your trip to maximize your chances of staying active.

33.12.2 Graph Traversals. A graph is a network connecting nodes together, with each node holding a property or properties. A *graph traversal* is the process of visiting the nodes of the graph in a particular order. Graph traversals are necessary to understand graph machine learning algorithms, for example, graph convolutional networks. They are also useful for understanding search algorithms applied to graph databases, as well as probabilistic graphical models such as Markov Chains and Bayesian Networks. The two major types of graph traversals are depth-first search (DFS) and breadth-first search (BFS). Depth-first search is a recursive approach to traversals where depth is prioritized over breadth, whereas breadth-first search is a level-by-level approach to traversals where breadth is prioritized over depth. An example of these traversal types is given in Figure 16. The DFS traversal of the graph in Figure 16 is `[A, B, D, G, E, C, F]` and its BFS traversal is `[A, B, C, D, E, F, G]`.

are studied to result in high performing neural network architectures, and there are some tips you should follow in order to design your neural network for a particular class of machine learning task.

The table below contains some of the most common activation functions used in neural networks:

Activation Function	**Formula**
Linear	$\sigma(x) = x$
Binary Step	$\sigma(x) = 1$ if $x \geq 0$, 0 otherwise
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$
ReLU	$\sigma(x) = max(0,x)$
tanh	$\sigma(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

The ReLU and tanh activation functions are typically used as the activation functions of nodes part of a hidden layer, and not as the output activation function of the network. The following table contains machine learning tasks and the output activation functions that should be used in a neural network architecture:

Task	**Output Activation Function**
Binary Classification	Sigmoid
Multi-Class Classification	Softmax
Regression	Linear

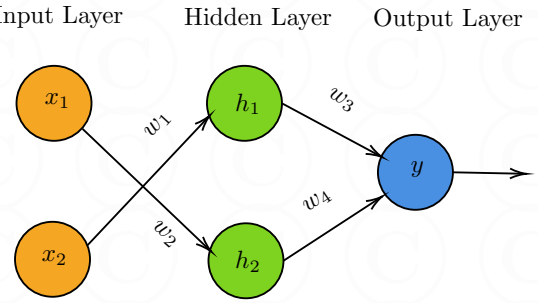
33.13.4 Gradient Descent (GD) Optimization. Recall that in a neural network, the model parameters are represented by the weights of the network. Gradient descent is an optimization strategy that is used to update the weights of a neural network. The concept of gradients is derived from calculus, where moving in the direction of the gradient vector is the direction of maximum function increase. Gradient descent attempts to minimize the total cost, it attempts to move weight vectors in the opposite direction of the gradient vector. This is the reason why in the gradient descent update rule, the weight at time $t+1$ is derived by subtracting the term $\alpha * \nabla J(\theta)$ from the weight at time t. Gradient descent optimization uses the following update rule:

$$w^{(t+1)} \leftarrow w^{(t)} - \alpha * \nabla J(\theta).$$

33.13.5 Training Strategies. (1) Early stopping is a technique that stops the training process as soon as the model's performance does not improve on the validation dataset.

35.5 A Concrete Example

Consider the following neural network with 5 nodes and 3 layers (1 input layer, 1 hidden layer, 1 output layer):



35.5.1 Forward Propagation. After the first epoch of the training process, the output of the neural network is defined as follows, given activation functions σ_1 for hidden node h_1 and σ_2 for hidden node h_2.

Recall that the output of a node is equal to the sum of the weights of the incoming edges to the node multiplied by the output of the outgoing node. We will use this to define the output of hidden layers h_1 and h_2 as follows:

$$h_1 = \sigma_1(w_1 x_2), \quad h_2 = \sigma_2(w_2 x_1) \tag{22}$$

The output of the neural network after the output activation is applied, y_{out}, is defined in terms of h_1 and h_2 as follows:

$$y = w_3 h_1 + w_4 h_2, \quad y_{out} = \sigma_3(y) \tag{23}$$

35.5.2 Backpropagation. As defined above, backpropogation is the process of updating the weights of a neural network. The initial weights of the neural network above are defined in the following table:

Weight	Initial Value
w_1	2
w_2	-2
w_3	4
w_4	1

The activation functions σ_1, σ_2, and σ_3 are defined in the following table:

Activation	Function
σ_1	$\sigma_1(x)=x$
σ_2	$\sigma_2(x)=\textbf{relu}(x)=max(0,x)$
σ_3	$\sigma_3(x)=\textbf{sigmoid}(x)$

The neural network is trained on the following dataset with a learning rate $\alpha=0.05$, and takes in each data point as input in the order in which the points are listed in the following dictionary: {`(2,2):1, (3,-1):0, (1,4):1`}.

After Pass #1 (First Epoch, First Instance):
In this step, we examine the final weights after the feedforward stage on instance (2,2) has been completed for the first epoch. Recall the neural network parameter update rule defined above:

$$w^{(t+1)} \leftarrow w^{(t)} - \alpha * \nabla J(\theta).$$

We are given weights $w^{(0)}$ for all weights of the network. In order to determine $w^{(1)}$ for all weights of the network, we need to compute the cost function, $J(\theta)$.

$$J(\theta)=\frac{\partial E_{net}}{\partial w}, \text{ where}$$

$$E_{net}=\frac{1}{2}\sum_{i=1}^{n}(\hat{y_i}-y_i)^2 \text{ and } n \text{ is the number of output nodes.}$$

In this case, since we only have one output node y, $E_{net}=\frac{1}{2}(\hat{y}-y)^2$. Expanding $\frac{\partial E_{net}}{\partial w_3}$ by applying the chain rule, weight w_3 in the neural network at time $t=1$ (after a single update) is as follows:

$$w_3^{(1)}=w_3^{(0)}-\alpha*\left(\frac{\partial E_{net}}{\partial y_{out}}\cdot\frac{\partial y_{out}}{\partial y}\cdot\frac{\partial y}{\partial w_3}\right).$$

The above update rule simplifies to the following:

$$w_3^{(1)}=4-0.05*\left((\hat{y}-y)\cdot\frac{\partial y_{out}}{\partial y}\cdot\frac{\partial y}{\partial w_3}\right)=4-0.05*((1-1)\cdot\ \ldots)=\boxed{4}.$$

This process is repeated for every weight of the neural network until all epochs have expired.

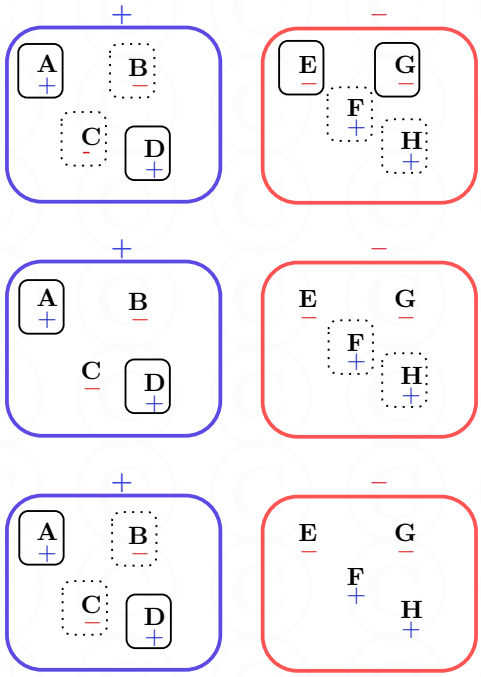


Figure 18
Accuracy, Precision, and Recall Metrics (top to bottom)

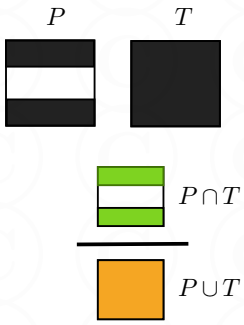
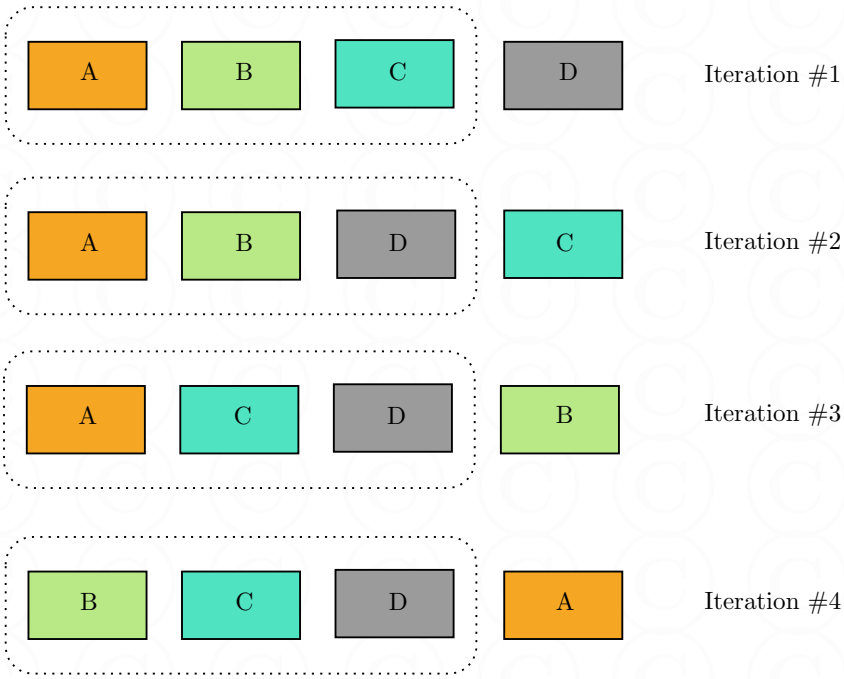


Figure 19
Intersection over Union (IoU)

validation error of the k rotations of the validation set. We describe this process in the following figure:



The boxes with labels from A to D are the training instances (rows of the training dataset). Note that in the case presented in the figure above, $k=4$. This is because one instance can be held-out from the training dataset as a validation instance. Each iteration in the figure above is referred to as a fold. Through a combinatorial perspective, there are a total of $\binom{4}{1}=4$ folds.

k-Fold Cross Validation

In k-fold cross validation, the value of k is equal to the number of instances in the training dataset.

36.6 Confusion Matrices

A confusion matrix is a visual that describes the performance of a classification model. A confusion matrix represents the number of data

38. Classification and Regression in Python Usage

In this section, we will cover the usage of the different libraries for both regression and classification tasks. Classification is a key sector of machine learning that aims to classify new data within one of many categories. Regression is another key sector of machine learning that aims to predict continuous data.

38.1 Linear Regression

For demonstration purposes, we will load the `Iris` dataset and analyze whether a relationship exists between the sepal length and the sepal width.

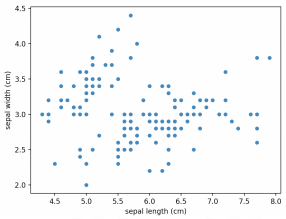
Step 1: Load the relevant `Iris` data.

```
from sklearn.datasets import load_iris
import pandas as pd
iris = load_iris()
data = pd.DataFrame(data=iris['data'],
    columns=iris['feature_names'])
```

Step 2: Import the `LinearRegression` library from `sklearn`.

```
from sklearn.linear_model import LinearRegression
```

Step 3: Visualize the data with a scatterplot.



Step 4: Define an untrained `LinearRegression` object.

```
lin_model = LinearRegression()
```

Step 5: Select the relevant columns from the `Iris` dataset to be trained on.

```
X = data[['sepal length (cm)']]
y = data['sepal width (cm)']
```

Step 6: Train the regression model by fitting the model with the relevant columns.

```
lin_model.fit(X, y)
```

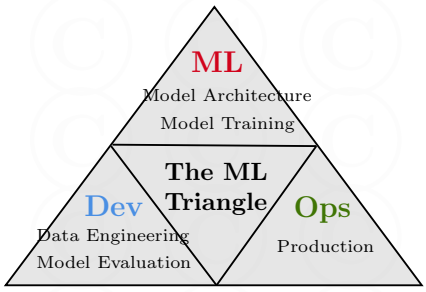
39. Model Deployment

Model deployment is the process of 'selling' a trained model for use. Before a model is deployed, it should be evaluated on a held-out validation or a test dataset to get a sense of how the model performs before it is used to make predictions in the real-world.

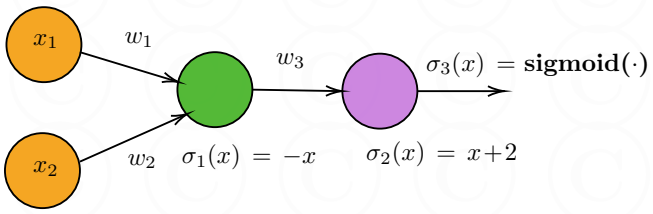
Consider a web application that allows users to input images or text and applies a machine learning algorithm on it. In this case, since the machine learning model is accessible to the public, it is considered to already be in the deployment stage. The designer(s) of the web application might have constructed and trained a deep learning model using TensorFlow, PyTorch, or Keras. However, since the model is locally stored as an object within one of the deep learning libraries above, the model could only be applied to data that is locally stored. In order to 'download' the model for a wider range of use, the model must first be prepared by downloading the model as a file that contains the model architectures and parameters. Such file types include `.json`, `.yaml`, and `.h5`.

Model preparation is the first stage of model deployment. A web framework such as Flask (Python) could have helped the web designer deploy the model, but before that, the model must be exported in order for the web framework to use it.

There are number of front-end platforms that are used to manage models and deploy them, including Amazon SageMaker, SAS, etc. These front-end platforms tend to follow a set of general efficient practices for maintaining and deploying machine learning models known as MLOps. If you are familiar with DevOps and the Agile methodology in Software Development, MLOps is analogous to them in the field of AI. The MLOps methodology consists of three stages: (1) the design stage, (2) the development stage, and (3) the operations stage.



3. Consider the following neural network with 1 hidden layer designed for a binary classification problem with weights w_1, w_2, and w_3 and activation functions σ_1, σ_2, and σ_3.

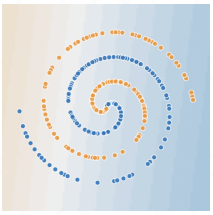


(a) The neural network is trained on a dataset with the data points (1,1) and (2,0) and their respective labels 1 and 0. The following table states the initial weights of the network as a result of a random weight initialization policy and the weights after n epochs of training the network on the data with gradient descent.

		Weights		
		w_1	w_2	w_3
Epoch	0	2	-1	1
	n	α	β	γ

True or False: For any choice of loss function $\mathcal{L}$, if $\gamma < 1$, then $\alpha < 2$, since the neural network parameters move in the same direction during each epoch of the training process. **True** **False**

(b) The neural network above is trained on the following data with the same weight initialization as in part **(a)**. After training the network for 100 epochs, the training loss remains exactly the same as the training loss after 1000 epochs. Briefly explain why this phenomenon occurs.

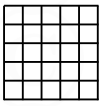


Solution. (a) False, because the parameters do not always move in the same direction; **(b)** this phenomenon occurs due to the linear activation functions σ_1 and σ_2 causing the neural network to be unable to learn non linearly separable data.

41. Deep Learning Foundations: Tensors and Convolutions

41.1 Tensors

Tensors and convolutions are the foundations of deep learning frameworks and algorithms. Convolutions are operations that are performed on tensors, and a convolutional layer consisting of convolution operations forms one of the many layers of a deep convolutional neural network. Tensors are higher-dimensional objects that are extensions to arrays and vectors. The concept of tensors came into fruition when convolutional neural network architectures were being developed and needed a three or higher dimensional input to be passed in to the first layer of the network. For instance, consider the simple two-dimensional tensor of shape 5×5, which is essentially a two-dimensional array/matrix you probably have already seen.



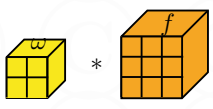
The following is a three-dimensional tensor, with length, width, and height components. The shape of the following tensor is $3 \times 3 \times 3$.

In deep learning, tensors are used to represent images of any dimension — a two-dimensional tensor might be used to represent an image classification problem on grayscale images, and a three-dimensional tensor might be used to represent an image classification problem on RGB images.

41.2 Convolutions

Convolutions are operations that may be performed on tensors. For a kernel (filter) ω and a tensor f with the same dimensionality, the kernel convolved with the tensor is denoted by the notation $\omega * f$. This operation is mathematically defined as follows:

$$\omega * f(x,y) = \sum_{i=-\alpha}^{\alpha} \sum_{j=-\beta}^{\beta} \omega(i,j) \cdot f(x+i, y+j) \tag{25}$$

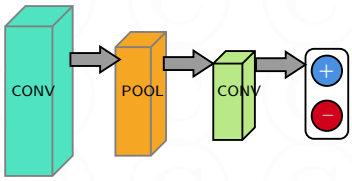


42. Deep Learning Architectures

The three arguably most popular deep learning architectures are the Convolutional Neural Network (CNN), the Recurrent Neural Network (RNN), and the Generative Adversarial Network (GAN). Each one of these architectures is suited for a particular type of task. For example, CNNs are used for image classification and segmentation tasks, RNNs are used for speech processing applications, and GANs are used for image, video, and voice generation. GANs will not be discussed in this section, but are discussed in Section 41: Advanced Deep Learning for Computer Vision.

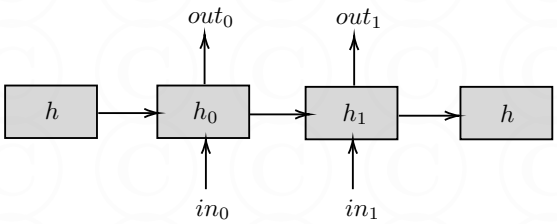
42.1 Convolutional Neural Networks (CNNs)

The CNN is a deep learning architecture consisting of a mix of convolutional (CONV), pooling (POOL), and fully-connected (FC) layers. The convolutional layers serve to downsample images, whose purpose is to extract particular desired features. from the image. The purpose of the pooling layers is to downsample feature maps by reducing their dimensionality. The fully-connected layers essentially simulate feed-forward networks networks where each neuron that is part of layer l is connected to all neurons that are part of layer $l+1$. The purpose of this layer is to allow for increased signal mixing, extending the CNN's locations of focus to a larger portion of the image.



42.2 Recurrent Neural Networks (RNNs)

RNNs are useful for speech processing and other Natural Language Processing (NLP) applications, which CNNs are not designed for. RNNs are designed for analyzing sequential or temporal data. In a RNN, 'hidden' modules are connected to each other, with each taking in an input from a sequence of data and producing an output.



43. Deep Learning for Image Classification

Consider the image below taken from Python's `skimage.data` library. The grayscaled form of this image is represented by a single-channel 2-D array (matrix) of intensities, with each entry representing a single intensity value for the corresponding pixel in the range [0,255].

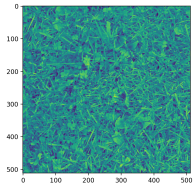


Figure 20
Grass, 512×512 RGB image

The following 2×2 matrix is the upper-left part of the 512×512 matrix of intensities representing the grayscaled form of the image above.

```
[[113, 114],[123, 115]]
```

Once the matrix of the intensities representing the entire image is obtained, classification can be performed by taking the features of the model to be the *flattened* vector of the intensity matrix. The flattened intensity vector of the sub-matrix above is as follows:

```
[113, 114, 123, 115]
```

After image normalization, this is in a form that is ready to be inputted as a feature vector.
Now, given a labeled set of training images, suppose we were to predict whether an image is one taken indoor or outdoor. With a set of images able to be represented as multiple feature vectors of intensities, they can be inputted into any model. The model we will use to describe the training process is a deep neural network. Each feature vector corresponding to a particular image will be inputted into the neural network one-by-one, and will be *forward-propagated* through the network. The output of the forward propagation will be compared with the true label to decide how the parameters of the network should be updated. The parameters of the network are then updated at a stage known as *back-propagation* of the training process.
This training process can be performed iteratively, with a `for`-loop, or with matrix operations, which tend to be more computationally efficient. For this reason, it is important to understand the basics of linear algebra, particularly matrix-vector operations.

45. Advanced Deep Learning for Computer Vision

45.1 Image Augmentation

Image augmentation is the process of increasing the size of the training data by adding transformed images of the original training data.

45.2 Object Detection

Object Detection is another computer vision technique that identifies objects within a particular image, placing bounding boxes around the identified objects. Deep learning models such as R-CNNs have been developed that achieve a high accuracy in performing such tasks.

45.3 Image Segmentation

Image Segmentation is a computer vision technique that breaks an image down into several segments. The most common types of image segmentation are semantic segmentation, thresholding segmentation, edge-based segmentation, and region-based segmentation. Deep learning models using the Convolutional Neural Network (CNN) architecture have been successful, from Oxford's VGG-16 model to Micrsoft's ResNet model. *Transfer learning* is the process of using a pre-trained model, such as either the VGG-16 or the ResNet model, to construct a full model architecture for a computer vision problem.

Figure 21
Object Detection (middle) and Image Segmentation (rightmost)

45.4 Neural Style Transfer (NST)

Neural Style Transfer, abbreviated as NST, is the process of inputting two separate images, one style image and one content image. The NST generates a new image by merging the style of the first image (discarding any content) with the content of the second image (discarding the background).

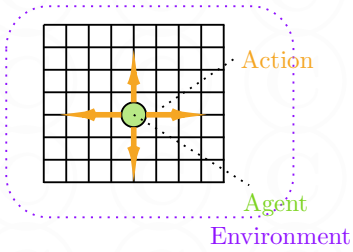
45.5 Generative Adversarial Networks (GANs)

Generative Adversarial Networks, abbreviated as GANs, are an unsupervised form of machine learning that aim to generate a new set of data having the same statistics as the data that was trained on. GANs consist of two components, a *generator* and a *discriminator*. The generator generates 'fake' data, whose output is then passed to the discriminator, whose job is to detect whether the data produced by the generator is 'fake' or not. This process continues until the generator is able to fool the discriminator, and the error rate of the discriminator increases.

48. Introduction to Reinforcement Learning

48.1 An Overview

Reinforcement learning is a form of machine learning that involves an agent learning from the environment it is placed within. The agent is placed in an environment that they learn from and aims to maximize their total reward, which is also referred to as utility. For example, an agent might be a self-driving car and the environment might be the places that the car passes through. As the car passes through different places, it learns different strategies, e.g., when to stop, when to switch lanes, etc.



48.2 The Exploration-Exploitation Tradeoff

A common theme in reinforcement learning is the tradeoff between exploration and exploitation. Exploration refers to taking risks, and exploitation refers to a continuation of past actions. An common classic reinforcement learning problem, the Multi-Armed Bandit problem, involves a player attempting to maximize their total reward (utility) after pulling arms. Each arm yields a certain amount of reward, and it is the player who decides which arms to pull in each trial to earn the maximum utility for themselves. In the Multi-Armed Bandit problem, exploitation would refer to the player pulling an arm that it has already pulled and received the optimal reward, and exploration would refer to the player exploring another arm that has not previously resulted in an optimal reward.

48.3 Q-Learning

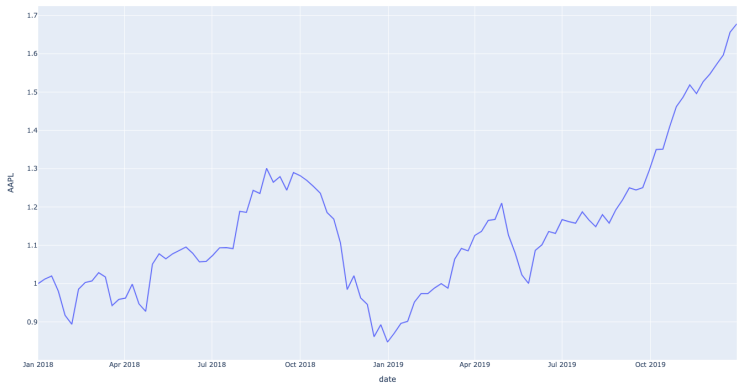
Q-Learning is a reinforcement learning algorithm that returns the optimal action to take, also known as the optimal policy, in a particular environmental state. For instance, a robotic vacuum using reinforcement learning might be at a corner of a room and may want to rely on a reinforcement learning algorithm such as Q-learning to decide which direction it should move in to pick up the most waste in the shortest period of time. Unlike supervised learning, Q-learning does not rely on a model getting updated as the agent learns from the environment, and rather maintains Q-values at different timesteps.

the Augmented Dickey-Fuller (ADF) Test, which is a hypothesis test with the following null and alternate hypotheses:

$$H_0: \text{ the time series plot is not stationary.}$$
$$H_a: \text{ the time series plot is stationary.}$$

49.4 Time Series Plots

The following is a time series plot of Apple's stock prices from January 2018 to January 2020 generated with Plotly, a front-end web interface for AI and ML operations:



The general trend of the time series plot above is positive, since there is a general increase in stock price from January 2018 to January 2020, even though there are some periods where the stock price falls. There appears to be irregular fluctuations in the time series plot above, which are fluctuations that cannot be predicted. There does not appear to be any regular seasonal or cyclic behavior taking place.

Given the time series plot above, you may want to forecast Apple's stock prices beyond January 2020. This may be useful in knowing when to buy or sell Apple stocks, since to maximize your profit, you would want to buy shares of stock when its price is low and sell shares of stock when its price is high.

To solve this, there are several time series forecasting methods available, including Autoregression (AR), Moving Average (MA), and Exponential Smoothing. The most used are the Autoregression and Moving Average models, but they should only be used for time series models without a trend or a seasonal component when decomposed.

50. A Deeper Dive into AI Systems

Below, we describe the technologies behind selected AI systems.

50.1 Facial Recognition Systems

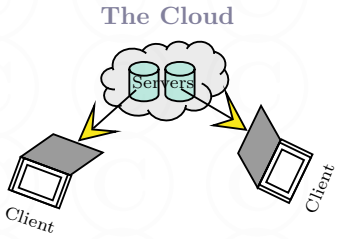
The facial recognition system is one of the most common AI technologies. The facial recognition system makes use of deep learning technologies, specifically, convolutional neural networks. The R-CNN scans and extracts all bounding boxes in an image by using a selective search strategy and checks whether it matches the face that is being searched for in the image with a classification model such as a Support Vector Machine (SVM) classifier. Researchers have developed faster versions of the R-CNN, called the Fast and Faster R-CNNs.

50.2 Recommender Systems

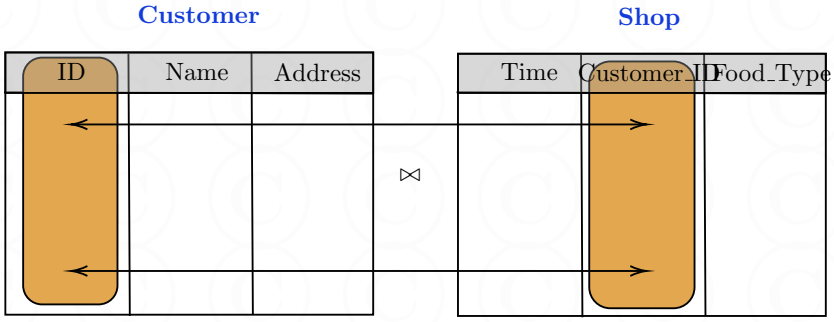
Recommendation engines are systems that suggest recommendations that match the preferences of users. For instance, when shopping on Amazon, you may have noticed that Amazon gives shoppers personalized items similar to the ones they viewed or purchased. Recommendation engines can be divided into content-based recommendation engines, collaborative filtering engines, or hybrid systems. Content-based recommendation engines recommends an item based on knowledge of the item, whereas collaborative filtering does not need information on the item, and only on the user's past decisions. Recommendation engines, like facial recognition systems, also tend to make use of deep learning technologies.

50.3 Cloud Computing Systems

Cloud computing systems, from computer systems to networks, also make use of AI. AI enables cloud systems to manage themselves, rather than engineers having to manage them. Companies such as Microsoft and Amazon provide specialized cloud services — Microsoft Azure and Amazon AWS respectively. Note that both cloud platforms provide on-cloud machine learning computing resources for customers as well, just like they make on-cloud SQL and NoSQL database systems available for customers to use.



The `JOIN` clause is another operator used to join two tables together on one semantically-matching column per table. We will describe how joins work below.



The following query retrieves the complete customer information, the time of purchase, and the type of food item purchased only for customers who purchased non-vegetable food items from the shop.

```
SELECT * FROM Shop
WHERE Food_Type = "Vegetable"
JOIN Customer ON Shop.Customer_ID = Customer.ID;
```

There are three types of join: (1) inner joins (⋈), (2) left outer joins (⟕), and (3) right outer joins (⟖). Inner joins only join rows of the relations being joined with matching values in the join columns, and not including the rows in both relations with non-matching values in the join column. However, in a left outer join, all rows in the left outer relation being joined will be included in the output relation (the relation after the join operation). Analogously, in a right outer join, all rows in the right outer relation being joined will be included in the output relation.

The `GROUP BY` clause is an operator used to separate or group a database relation based on a particular field.

ID	Name	Price
101	Orange	Low
102	Strawberry	Medium
103	Blueberry	High
104	Raspberry	High

The `HAVING` clause is similar to the `WHERE` clause, except that it checks for a condition on a group in a dataset rather than on an entire dataset. For example, going back to the example above with the three price groups, if we group by price and place a `HAVING` clause after that, we would get an output dataset with one row per group.

Consider the following SQL query that runs on the relation `Fruits` storing fruits and their prices above:

```
SELECT F1.Price, COUNT(*) FROM Fruits F1
GROUP BY F1.Price
HAVING (SELECT COUNT(*) FROM Fruits F2 WHERE F2.Price = F1.Price
    AND F2.Name = "Orange") > 1;
```

The query above selects the number of fruits for each price range that contains at least two oranges. In this case, we grouped on the price of the fruits, and in our output relation, we had at most one row per price range. The output relation of running the query above on the `Fruits` relation above is empty, i.e., contains 0 rows, since there were no price ranges with two or more oranges classified within that price range.

Now, we will turn to the SQL Data Definition Language (DDL). We define the syntax of the SQL DDL below:

- `CREATE TABLE [TABLENAME]` – creates a table with name `TABLENAME`.
- `PRIMARY KEY` – identifies a column as a primary key of its corresponding table.
- `FOREIGN KEY ([FKCOL]) REFERENCES [TABLENAME] ([FKCOL2])` – identifies column `FKCOL` as a foreign key of its corresponding table that references logically identical column `FKCOL2` in table `TABLENAME`.

We will get into primary and foreign keys in a future section, but we will provide a brief definition below:

Database Primary and Foreign Keys

A **primary key** is a field that uniquely identifies a table. A **foreign key** is a field that references a logically connected field of another table.

51.1.8 NoSQL (Non-Relational Databases). NoSQL stands for "Not only SQL". In practice, NoSQL is also referred to as a non-relational database.

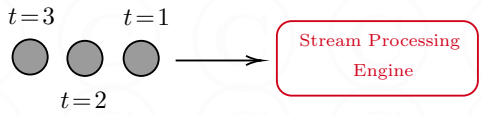
A non-relational database is one that does not use the tabular scheme that relational databases use. Rather, they utilize a storage model that is optimized for the specific requirements of the type of data that is stored. Popular non-relational databases are MongoDB, DocumentDB, Cassandra, and a few others. Non-relational databases work with semi-structured data, such as data stored in XML or JSON files. Common classes of non-relational databases include graph data, stream data, and spatial data.

The following table compares relational databases with non-relational databases:

Relational Database	Non-Relational Database
Simple	Simple Design
Robust	Fast
Flexible	Strong control over availability
Scalable	Better "horizontal scaling" to clusters of machines

Table 4
Relational vs. Non-Relational Databases

Stream data refers to data that continuously enters a data platform in timestamp order. Stream data processing deviates from the standard batch form of data processing where the data is processed once all data has been loaded into a platform. Apache Kafka and ksqlDB are distributed stream database systems specialized for stream data.

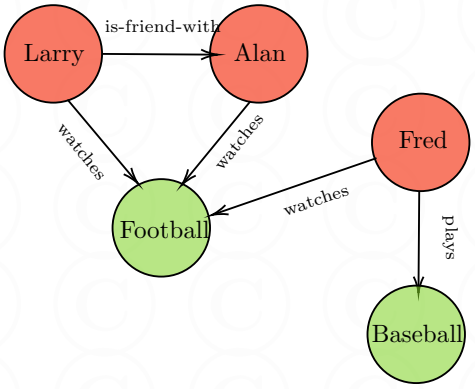


Spatial data refers to data that holds geographic properties. There are specialized database management systems that provide specific functionalities for spatial data such as quad trees for indexing.

Graph databases are forms of non-relational databases that are classified as NoSQL databases and are suited for data operations on connected data, such as friend connection data or the world wide web network. Some common graph database management systems include

Neo4j, ArangoDB, and CosmosDB. Each one of these have their own language that they accept (similar to SQL) — for instance, Neo4j uses the Cipher language, and CosmosDB uses the Cassandra Query Language (CQL). Graph database systems place ACID guarantees on transactions similar to relational database systems.

The following graph represents connections between people and types of sports:



As with all graphs, the graph above contains a set of nodes and a set of edges. The nodes represent objects, both people and the types of sports, and the edges connecting two or more nodes together represent the links between the nodes. For instance, some edges in the graph above have a label of 'watches' that connect the outgoing node representing the person and the incoming node representing the sport. This edge defines the relationship that the person watches the sport. Without the edge between the person and the sport, this relationship would not be encoded in the graph.

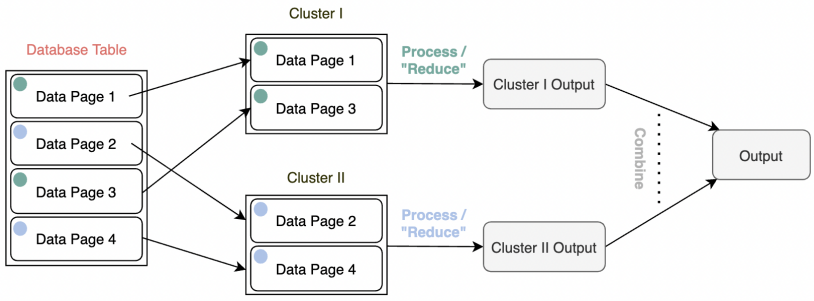
51.1.9 Database Design & Schema. This section will aim to answer the question of how databases are usually constructed? As defined previously, databases are collections of tables. However, would it really make sense to put together tables with unrelated information within a single database object. As you might have expected, the answer is no. A database should be carefully organized, with the tables of the database grouped in such a way such that they are **logically connected**. From a logical connection of tables, we can construct a design, similar to the common object-oriented design for constructing classes, known as a *database schema*.

SaaS	PaaS	IaaS
Easy to use & setup Can lead to website outages	Good control over platform software Can only control what is built on the platform	Offers maximum control One has to ensure it runs properly

Table 6
Pros and Cons of SaaS, PaaS, & IaaS

51.4.6 Motivation for Distributed Computing. Thus far, we have not been considering the amount of compute power available for use. A problem arises when we do not have enough compute power to process a sequence of transactions on an entire database or a large-part of it. To resolve this, companies such as Google, Amazon, and Facebook, who have many requests to their databases have to turn to a framework known as *distributed computing*. Distributed computing allows for concurrent/parallel operations, where transactions can be executed at the same time on different chunks of the database. In the section below, we will delve into the terminology and the high-level distributed computing architecture.

51.4.7 A Snapshot of the Distributed Computing Model. *Distributed Computing* for database systems is a model that improves standard processing performance in which an operation on a large dataset is split into several sub-operations on small chunks of the dataset. The results from the sub-operations are then combined to produce the final result. This is the general process of MapReduce, which is the principal processing component of Apache Hadoop.

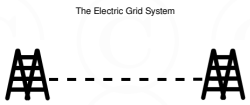


51.5 References

1. "The Relational Database." Amazon, Chapman & Hall, 1995. aws.amazon.com/relational-database/.
2. "What Is PaaS? Platform as a Service: Microsoft Azure." Platform as a Service — Microsoft Azure, Microsoft, azure.microsoft.com/en-us/overview/whatis-paas/.

53. Application: Data Analysis on Power Systems

An energy company serves customers by providing them with power. They trained an long short-term memory (LSTM) model on energy data from the past two years, and predicted that less than 10% of the customers would be *affected* for the current summer months. A customer is said to be *affected* for a particular month if they experienced more than 5 hours of outages in the month. However, it turned out that several massive power outages hit the residents of the city in July and August 2021, causing a power blackout requiring attention. Because of their prediction, the energy company planned little in advance for such an event. They ask your team, *Data Quality*, to investigate the issue by looking into the model used by the *Analytics* team, and ask you to update the databases with the latest unrecorded observations.



It turns out that the Analytics team did not communicate with your Data Quality team for the 2020 data before they began their analysis, a big contributor to the prediction error for the summer. Your team discusses preparing for the winter by cleaning the database, `PowerOutages2020.accdb`, and sending the updated version over to the Analytics team.

Month	Year	Avg. Energy Consumption	Number of Customers Affected
Jan	2020	1600 kWh	12500
Feb	2020	1250 kWh	10000
Mar	2020	1400 kWh	11000
Mar	2020	1440 kWh	11082
⋮	⋮	⋮	⋮
Jun	2020		13000
⋮	⋮	⋮	⋮
Dec	2020	1300 kWh	9500

Table 7
`PowerOutages2020.accdb` **(the uncleaned data analyzed by the Analytics team)**

Which of the following describes the possible reason(s) behind the analysis on the partially uncleaned data resulting in an inaccurate prediction for the summer? Select all that apply:
A. The system collecting energy usage data was under repair last summer and the missing data was filled in from the data gathered from the previous summer.
B. All individuals affected from the same household, normally considered as separate customers affected, were considered as a single customer affected only in 2020.
C. A seasonal storm hit the region with greater impact relative to storms in other seasons.
D. The onset of COVID-19 led to two unlabeled records for March 2020, one intended for the pre-COVID days and one for the official COVID days.
Answers: *(A), (B), (D).*

Index

k-Fold Cross Validation, 146
k-Nearest Neighbors (kNN), 127

Absolute Error, 120
ACID Properties of Database Systems, 168
Activation Functions, 137
Agglomerative Hierarchical Clustering, 149

Bagging, 132
Bar Charts, 37
Bayes' Theorem, 82
Bayesian Statistics, 82
Bias in Machine Learning, 118
Bias-Variance Tradeoff, 118
Bivariate Distributions, 64
Boosting, 132
Boxplots, 38

Chain Rule for Derivatives, 88
Chebyshev's Inequality, 67
Classification, 116
Classification and Regression in Python, 150
Cloud Computing, 183
Cloud Computing Models, 184
Cloud Computing Systems, 166
Cloud vs. On-premise Environments, 184
Cluster Analysis, 149
Clustering, 117
Combinations, 71
Combinatorics, 70
Confidence Intervals, 81
Continuous Random Variables, 62
Convolutional Neural Networks (CNNs), 158
Convolutions, 157
Correlation, 64, 80
Correlation and Covariance, 51
Covariance, 64, 81
Cross-Entropy Loss, 120

Data Analysis on Financial Investments, 189
Data Analysis on Power Systems, 188
Data Partitioning, 42
Data Preprocessing, 32, 35
Data Structures and Algorithms, 93
Database Design & Schema, 177
Database Normalization, 179
Databases, 167
Databases and Cloud Computing, 167
DataFrames, 45
Decision Trees, 125
Deep Learning, 139
Deep Learning for Action Recognition, 160
Deep Learning for Image Classification, 159
Deep Learning for Text Mining, 162
Density Functions, 51, 56
Derivatives, 88
Descriptive Statistics, 80
Diagonal Matrices, 91
Dimensionality Reduction, 36
Discrete Random Variables, 60
Distribution Quantiles, 53
Distributions, 52
Divisive Hierarchical Clustering, 149

Dot Product, 90

Efficiency and Big-O Notation, 97
Efficiency in Database Operations, 180
Eigenvectors and Eigenvalues, 92
Empirical Risk Minimization (ERM), 146
Ensemble Models, 132
Entity Relationship Diagrams (ERDs), 178
Epoch, 32
Estimators, 67
Events, 50
Expectation, 50, 51
Exploratory Data Analysis, 40

Facial Recognition Systems, 166
Features, 32
Field Normalization, 35
Fully-Connected Neural Networks, 137
Functional Dependencies, 179
Functional Programming for Data Analytics, 190
Functions of Random Variables, 60

Gaussian Mixture Models (GMMs), 54
Generative Adversarial Networks (GANs), 161
Gradient Descent, 138
Graph Traversals, 135
Graphs, 94

Handling Missing Values, 35
Heatmaps, 39
Histograms, 37
Hyperparameters, 121
Hypothesis Testing, 68, 81

Identity Matrices, 91
Image Segmentation, 161
Independence, 55
Inner Join, 170
Instance, 32

K-Means Clustering, 126
Keras, 47

Labels, 32
Left Outer Join, 170
Line Plots, 38
Linear Classifiers, 128
Linear Discriminant Analysis (LDA), 131
Linear Regression, 122
Linked Lists, 94
Lists, 93
Logistic Regression, 124
Loss Functions, 120
Lower and Upper Triangular Matrices, 91

Machine Learning Class Distributions, 119
Markov's Inequality, 67
Matplotlib, 47
Matrices, 90
Matrix Inverses, 92
Maximum Likelihood Estimation (MLE), 75
Mergesort and Graph Traversals, 98
Min-Max Normalization, 35
ML Model, 32
ML Trade-offs, 34
Model Deployment, 34, 153
Model Evaluation, 34, 143
Model Training, 32

Neural Networks, 136
Neural Style Transfer (NST), 161
Normal (Gaussian) Distributions, 53
NoSQL (Non-Relational Databases), 176
NumPy, 44

Object Detection, 161
One-Hot Encoding, 36
Ordinal Encoding, 35
Ordinary Least Squares (OLS), 65
Outlier Analysis, 149
Overfitting, 33

Pandas, 45
Partial Derivatives, 88
Pearson Correlation, 66
Permutations, 72

Pie Charts, 37
Precision, 144
Predicted Label, 32
Predictor Variable, 32
Probability, 50
Python for Data Analysis, 43
PyTorch, 47

Queries, 168
Queues, 93

Random Variables, 50, 56
Recall, 144
Recommender Systems, 166
Recurrent Neural Networks (RNNs), 158
Recursive Call Stack, 73
Recursive Counting, 72
Regression, 116
Reinforcement Learning, 163
Relational Databases, 168
Relational vs. Non-Relational Databases, 176
Response Variable, 32
Right Outer Join, 170

Sample and Population Statistics, 52
Scatter Plots, 38
SciPy, 47
Segmentation, 117
Series Arrays, 46
Set Notation, 51
SkLearn, 47
Splitting Ratio, 42
SQL, 169
SQL DDL, 171
Squared Error, 120
Stacks, 93
Stationary Time Series Data, 164
Support Vector Machines, 129

TensorFlow, 47
Tensors, 157
The Bernoulli Random Variable, 60
The Binomial Random Variable, 60
The Cauchy Distribution, 63
The Central Limit Theorem, 67
The Chi-Square (χ^2) Test, 69
The Coefficient of Determination (R^2) Metric, 66
The Data Pipeline, 31
The Derivative Operator, 49
The Exponential Random Variable, 62
The Gamma Random Variable, 62
The Geometric Random Variable, 61
The Hidden Markov Model (HMM), 134
The Integral Operator, 49
The Law of Large Numbers, 67
The Naïve Bayes Classifier, 133
The Negative Binomial Random Variable, 61
The Normal Distribution, 62
The Perceptron, 129
The Poisson Random Variable, 61
The Product Rule, 70
The Sigma Operator, 49
The Sum Rule, 71
Time Series Data, 164
Time Series Data Analysis, 164
Time Series Decomposition, 164
Tree and Hash Indices, 180
Trees, 94
True Label, 32
Type I and Type II Errors, 69

Underfitting, 34

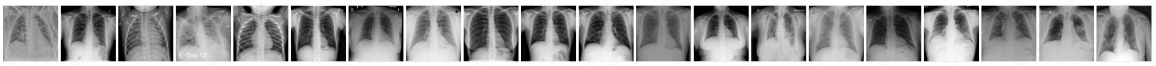
Variance, 50, 51
Vectors, 89

Part II: Case Studies

Case Study VI: COVID-19 Chest XRay Screening

In this case study, we will screen whether patients have COVID-19 or not given a snapshot of their 2D Chest XRay image. The motivation for using this prototype in the medical industry is that it tends to be faster in screening patients than most radiologists, and may also be more accurate as it can detect subtle differences in images that radiologists might miss. And, especially in the healthcare industry, the speed and accuracy of results are critical, as the cost of misdiagnosing a patient might have severe effects. We will implement a deep learning architecture with transfer learning in Keras, train it on a collection of 2D Chest XRay images for both COVID-19 positive and negative patients, and test our results on a held-out validation set.

[27]:
```
test_imgs = np.zeros((1,256,256,3))
fig, axs = plt.subplots(1, 20, figsize=(256,256))
cnt = 0
for images, labels in val_ds.take(2):
    for i in range(10):
        test_img = images[i].numpy().astype("uint8")
        test_label = val_ds.class_names[labels[i]]
        if cnt < 10:
          axs[i].set_title(test_label)
          axs[i].imshow(test_img)
        else:
          axs[i+10].set_title(test_label)
          axs[i+10].imshow(test_img)
        test_img = test_img.reshape((1,) + test_img.shape)
        if cnt < 20: test_imgs = np.vstack((test_imgs, test_img))
        cnt += 1
test_imgs = test_imgs[1:]
fig.tight_layout()
plt.title('1 Batch of Validation Data')
plt.show()
```



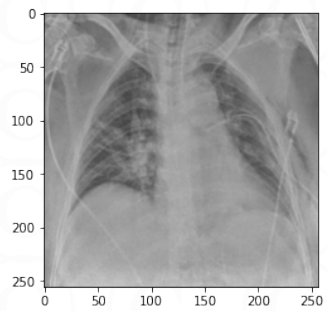
[28]:
```
test_imgs.shape
```

[28]:
```
(20, 256, 256, 3)
```

As shown above, the shape of each test image is (256, 256, 3); the image is of shape (256,256) and has 3 RGB channels. The 20 for the first dimension represents the 20 images that we are testing and displaying above.

[29]:
```
test_img = test_imgs[0,:,:,:]
test_img = test_img.astype("uint8").reshape((1,) + test_img.shape)
plt.imshow(test_img[0])
```

[29]:
```
<matplotlib.image.AxesImage at 0x7fac744af510>
```



Further Case Study Analysis

1 Background

In medical imaging, radiologists have to take scans of patients and analyze them to produce an output. With the advent of AI and deep learning, this process can be automated, saving manual workload, improving efficiency and results in precise decisions, and enabling speedy delivery of results to patients.

Medical imaging is most commonly studied in detecting pneumonia from patient X-rays. COVID-19, like Pneumonia, is studied to have a strong correlation with a patients' chest X-ray images. Because of this, we are able to develop a model that we know that will achieve a reasonably high accuracy when given a new chest X-ray image. This system will make radiologists' job easier in classifying patients' X-ray images, and will likely make the process faster.

Our case study is a machine learning classification task with the help of a **deep learning model** that takes in as input an X-ray image of a patient's chest and predicts whether the patient is positive or negative for COVID-19. Since there are two output classes of prediction, this is a binary classification problem.

For this case study, we used Python as our primary programming language, and we used the GPU built in to Google Colab in order to speed up the training and testing processes. The following are the key libraries we used for the case study:

- **Deep Learning:** Keras, TensorFlow
- **Scientific Processing:** Scikit-Learn

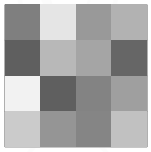
To construct the (convolutional) deep learning model architecture, we applied *transfer learning*, that is, importing an existing, pre-defined network architecture. We chose the most appropriate model, taking into consideration the number of layers of the network, and toggled hyperparameter values to strive towards optimality.

To display the classification results, we could design a web user interface with Flask (a built-in Python web development library), JavaScript, and HTML, that allows users (radiologists) to input an image of any dimension. The image would get sent over to the trained model, and the final classification result is to be returned. The final output to the user (radiologist) would be the output of the trained model.

2 Training and Testing Images

The images that we used to train and test the model were grayscale X-ray images, i.e., there is a single channel, and each pixel contains a property representing its intensity between 0 (black) and 255 (white). The following is a sample chest X-ray image that will be part of our training dataset:

(a) Sample X-Ray Image



(b) Sample Grayscale Image

20	120	50	80
10	85	76	15
225	10	22	78
100	48	24	90

(c) Pixel Matrix for **(b)**

The image is represented as a matrix of pixel intensities, with each element of the matrix ranging from 0 to 255.

3 Preprocessing Raw Data

The first step we performed was preprocessing the raw images. The images that are inputted by radiologists could be of any dimension. However, since our model strictly accepts 256×256 dimensional images, the image needs to be reshaped to a 256×256 dimensional image that is able to be taken in by our model.

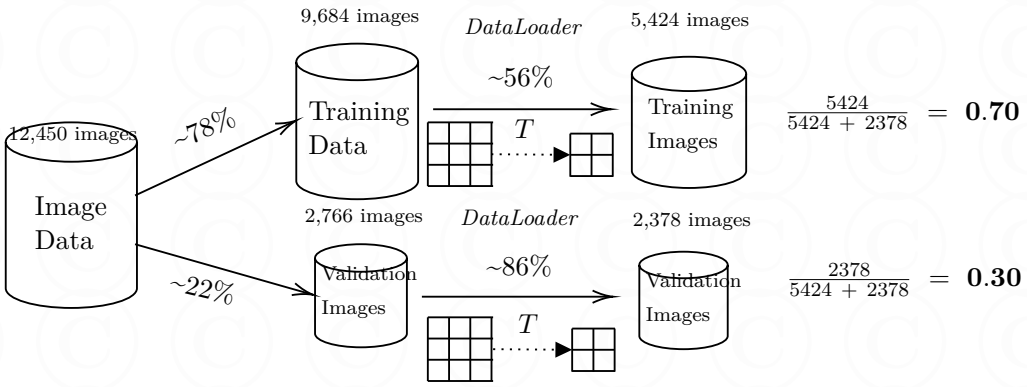
We also performed data augmentation where we performed reshaping, re-scaling, zooming, shifting, shearing, rotation, and flipping to each image of the training and validation datasets. We used Keras' `DataLoader` functionality for this, creating two `DataLoader`s, one exclusively for the training data and one exclusively for the validation data.

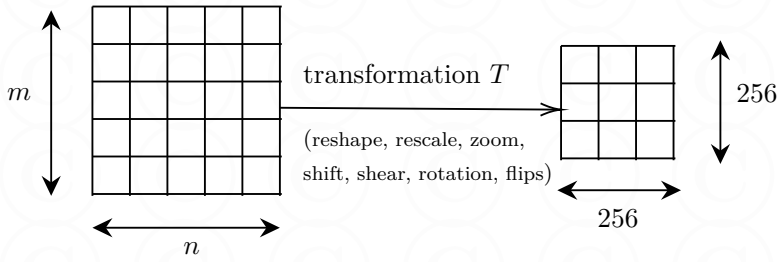
```
datagen_train = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.44,
    fill_mode='nearest'
)
```

```
datagen_val = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.86,
    fill_mode='nearest'
)
```

The purpose of this step was to reduce overfitting and make the model more generalizable and

recognizable to distortions in images.

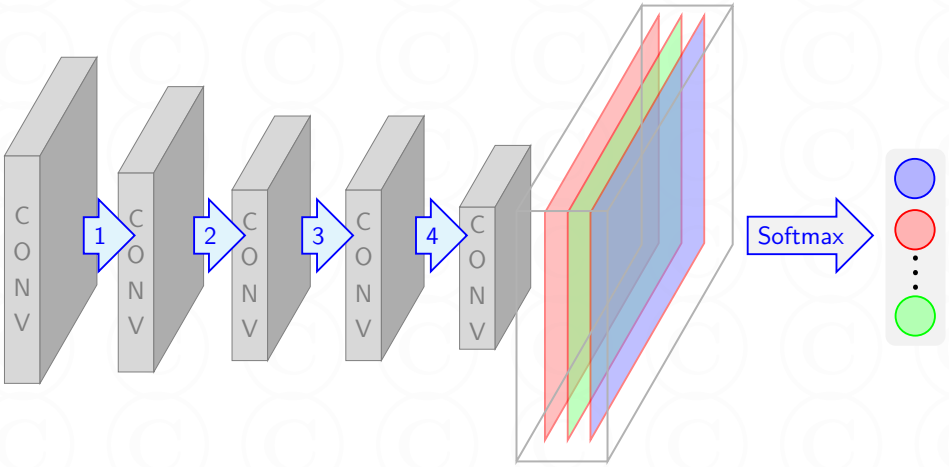




4 Deep Learning Architectures

Deep learning architectures are based upon neural networks. In our case of image classification, the features that are independently inputted into the network are the individual pixel intensities.

The deep learning architecture we used was a convolutional neural network, which tends to have a combination of convolutional and pooling layers.



4.1 Convolutions

Convolutional layers rely on individual convolution operations. The VGG-16 architecture uses same convolution and a 3×3 filter with a stride of 1. A *convolutional operator* '*' applied to a 2-dimensional image $f(x, y)$ and a filter ω performs the following operation:

$$\omega * f(x, y) = \sum_{i=-\alpha}^{\alpha} \sum_{j=-\beta}^{\beta} \omega(i, j) \cdot f(x + i, y + j) \tag{1}$$

The following is an example of the specific convolution the VGG-16 model uses, but with a lower resolution image. If a pixel in the output image is computed to have a negative intensity, we clamp it, i.e., we set it to 0.

$f(x, y)$

10	80	105	218
102	40	120	225
68	210	210	8
202	42	210	20

$*$

ω

1	-1	1
0	0	0
-1	1	-1

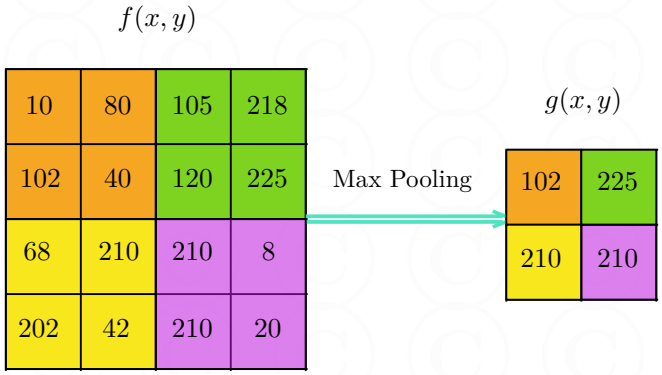
$=$

$g(x, y)$

62	0	0	105
0	0	185	0
0	0	293	0
142	0	0	0

4.2 Max Pooling

The max pooling layers of a VGG-16 architecture compute the maximum intensity for each patch and replace that patch with the computed maximum intensity. This reduces the dimensionality of the original image, creating a down-sampled feature map that smooths the features. The following figure illustrates the max-pooling process on a 4×4 image:



5 Constructing Our Deep Learning Architecture

After preprocessing the raw input images, we applied *transfer learning*, i.e., imported an existing deep learning model architecture. Specifically, we imported the VGG-16 architecture, which is a convolutional neural network (CNN) architecture developed at the University of Oxford. The VGG-16 model consists of 16 layers — a mix of 2-dimensional convolutional layers and 2-dimensional max pooling layers — and approximately 138 million parameters.

7.2 The F1 Score Metric

The F1 Score metric can be easily computed by performing an aggregate operation on the confusion metric. Because of this, we create the confusion matrix for the model applied to the validation dataset.

```
hard_preds_val = [1 if lbl > 0.50 else 0 for lbl in preds_val]
confusion_matrix(val_generator.classes, hard_preds_val)
```

Output Confusion Matrix:

```
array([[542, 79], [25, 1731]])
```

The confusion matrix above can be visualized easier in the following table:

	Predicted Normal	**Predicted COVID**
True Normal	542	79
True COVID	25	1731

Table 3: Confusion Matrix

The confusion matrix tells us that $542 + 1731 = 2273$ images from the validation set were correctly classified and $79 + 25 = 104$ were incorrectly classified.

$$\text{Validation Accuracy} = \frac{2273}{2273 + 104} = 0.956.$$

The validation accuracy computed above matches with the validation accuracy Python returned. What the confusion matrix does tell us, that the plain validation accuracy output does not indicate, is the exact number of true positives, true negatives, false positives, and false negatives. We compute the precision and recall statistics, as well as the F1 score below:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{542}{542 + 79} = 0.8728.$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{542}{542 + 25} = 0.9559.$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \cdot \frac{0.8728 \times 0.9559}{0.8728 + 0.9559} = \boxed{0.9125}.$$

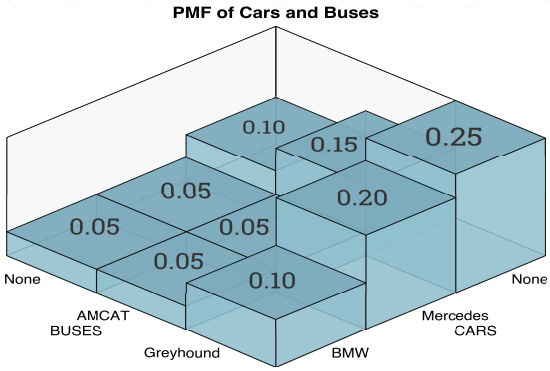
Since the F1 score ranges from 0 to 1, with higher scores indicating better performance, the F1 score of 0.9125 above is reasonably high.

No. Epochs	**Training F1 Score**	**Validation F1 Score**
5	0.8929	0.8734
10	0.9052	0.8805
20	0.9187	0.9125

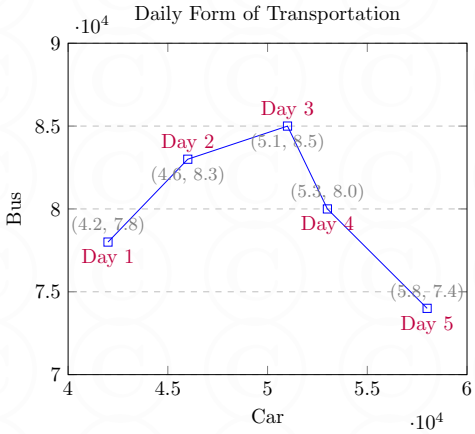
Table 4: Training and Validation F1 Scores for Different Epochs

Part III: Data Science and Analytics Skills Assessment (DSSA)

Questions 16-22 refer to the following scenario. The means of commute from home to work and vice versa for five days in a city are represented by three data visualization tools as follows for EDA: a PMF (Figure 1a), a line graph (Figure 1b), and a table (Figure 2). The PMF and the table count the number of people transported, whereas the line graph counts the number of car and bus ride occurrences. However, all three models take each person into account only on their first day arrival.



(a) PMF



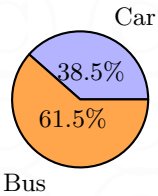
(b) Line Graph

		Buses		
		Greyhound	AMCAT	None
Cars	BMW	50,000	25,000	25,000
	Mercedes	100,000	25,000	25,000
	None	125,000	75,000	50,000

Figure 2: Table

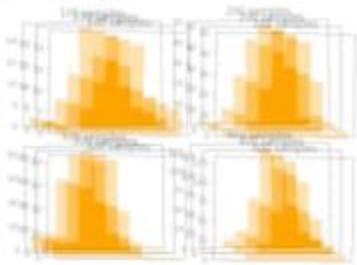
16. Learning Objective: Statistical Analysis.
(a) i. What is the probability that a randomly selected person is transported in a BMW car on their first day? ii. Similarly, what is the probability of transport in a Mercedes car and a Greyhound bus?
(b) Are the events represented by the cells in Figure 2 disjoint? Justify your answer.

17. Learning Objective: Model Comparisons. Is the following pie chart consistent with the line graph above in showing the cumulative shares of car and bus transportation occurrences counted over the five days?



18. Learning Objective: Linear Models. Let X denote the number of car transportation occurrences counted daily and Y denote that of bus transportation. Given that the line of best fit through Figure 1b is $\hat{y} = -0.253x + 92662$, where 'Car' is the predictor variable and 'Bus' is the response variable, what is the covariance between X and Y? Give an expression for the percent error, without absolute values, of the estimate given by the linear fit for a day with the same number of car and bus occurrences.

19. Learning Objective: General Data Analysis. *True or False.* (a) Since the data is categorical, we should use a Chi-Square Goodness of Fit Test to test whether the distribution of the sample data in the table matches that represented by the probability mass function. (b) As the number of cars counted increases daily, the number of people counted who ride buses and not cars could remain the same.



Answer:

A. low B. medium C. high D. Either low or medium

Answer:

25. **Learning Objective: Feature Mapping Perceptrons** A particular non-linear decision boundary is learned by extending the feature space with feature map ϕ: $\phi(x_1, x_2) = (x_1, x_2, x_3)$: $x_3 = x_1 - x_2$, and applying the Perceptron Learning Algorithm to it. Given the following trained Perceptron in the higher-dimensional space, which *of the following* represents the original non-linear decision boundary?

A. $x_2 = -2/3x_1$ B. $x_2 = -2/3x_1$ C. $x_2 = \pm\sqrt{x_1^2 + 2/3x_1}$ D. $x_2 = \pm\sqrt{x_1^2 - 3}$

Answer:

Questions 43-45 refer to the following code in Python. A *known-acid-unknown-base titration* is a chemical reaction between an unknown base and a known acid with a known concentration, measured by a pH indicator. A solution with pH = 7 is neutral, one with pH < 7 is acidic, and one with pH > 7 is basic. In this process, the known acid is added to the unknown base and neutralizes it, forming a solution with a pH = 7. The equivalence point of the titration is where the neutralization occurs, and can be seen by a change in color of the indicator. The following is a sample titration curve for an arbitrary acid and base, where a known base is added to an unknown acid, explaining the increase in pH as the volume increases:

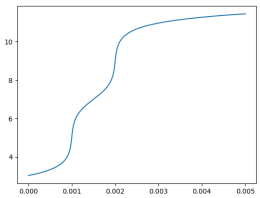


Figure 4: Sample Titration Curve

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.svm import SVR
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression

def op1(df,types):
    df = df[df['Acid Type'] == types[0]]
    df = df[df['Base Type'] == types[1]]
    return df

def op2(df,model,types,val):
    df = op1(df,types)
    X = pd.DataFrame(np.array(df['Volume']).reshape(-1,1))
    y = df['pH']
    model.fit(X,y)
    pred = model.predict([[val]])
    plt.plot(df['Volume'], df['pH'])
    plt.show()
    return [pred.item(), model.score(X,y)]

def op3(df,lr,types):
    df = op1(df,types)
    X = pd.DataFrame(np.array(df['pH']).reshape(-1,1))
    y = df['Volume']
    lr.fit(X,y)
    pred = lr.predict([[7.0]])
    return pred.item()

def op4(df,types):
    df = op1(df,types)
    X = pd.DataFrame(np.array(df['Volume']).reshape(-1,1))
    y = df['pH']
    clf = svm.SVR(kernel='linear', C=0.001).fit(X, y)
    return np.mean(cross_val_score(clf, X, y, cv=3, scoring='↘
```

46. **Learning Objective: Evaluating Loss Functions.** Consider a dataset containing complete NBA player information. From an unknown player's points, rebounds, and assists per game, a sports analyst, who has studied the dataset, predicts other information about the player, as listed below. Which of the following fits best with the 0-1 loss function f_0, which is defined as 0 when $h(\vec{x}) = y$ and 1 otherwise?
Hint: *for each option, consider the elements of the set of labels $\vec{y}$.*
A. Player Age B. Years of experience C. Position played D. Average steals per game
Answer:

47. **Learning Objective: Linear Algebra.** In the process of applying the Skip-Gram Word Embedding paradigm, we have a n-dimensional vector $\mathbf{a}$ and the following weight matrix $\mathbf{W}$. Which of the following transformations does left-multiplying $\mathbf{W}$ to $\mathbf{a}$ perform?

$$\mathbf{W}_{n \times n} = \begin{bmatrix} 2 & & & \\ & \ddots & & \\ & & \frac{1}{2} & \\ & & & -1 \end{bmatrix}$$

A. It reverses the elements of $\mathbf{a}$ and scales them by a factor of n.
B. It reverses the elements of $\mathbf{a}$ and scales them by a factor of $1/n$.
C. It scales the elements of $\mathbf{a}$ by a factor of n.
D. It cannot be determined, as the different versions of $\mathbf{W}$ each produce different results.
Answer:

48. **Learning Objective: Linear Algebra.** Singular Value Decomposition (SVD) is a technique commonly applied to a set of training data to reduce its dimensionality and reduce the storage consumption of the data. It factorizes an $m \times n$ matrix A into the form USV^T, where U is of $m \times k$ dimension, S is of $k \times k$ dimension, and V is of $n \times k$ dimension. The first p columns of matrix V in SVD are said to be the principal component vectors in PCA, and are orthogonal. Which of the following statements is correct regarding the process described above?
A. The principal component vectors are in the same directions as the eigenvectors of A.
B. The principal component vectors are in the same directions as the eigenvectors corresponding to the largest eigenvalues of AA^T or A^TA.
C. The principal component vectors have the same magnitudes as the eigenvalues of AA^T, not A^TA.
Answer:

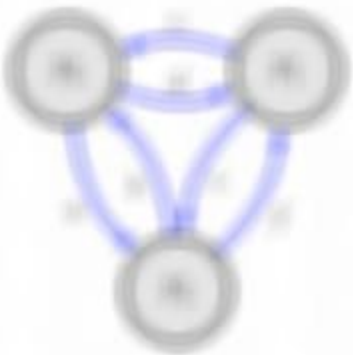
49. **Learning Objective: SQL Queries.** A company writes the following SQL query to assist them in their candidate search process. They maintain a database with tables `T1` and `T2`, where `T1` contains the names of candidates (`Name`), their IDs (`ID`), their years of experience (`YearsExp`), and an indicator variable representing whether they took a data mining course (`CourseTaken`), and `T2` contains only the names of candidates (`Name`). Which of the following operations does the code below perform?

```
SELECT T1.ID, T2.Name
LEFT JOIN T1 ON T2.Name = T1.Name
WHERE T1.YearsExp > 3 AND T1.CourseTaken = 1
INTERSECT
SELECT T1.ID, T2.Name
FROM T1 JOIN T2 ON T2.Name = T1.Name
WHERE T1.YearsExp > 3 AND T1.CourseTaken = 1
ORDER BY T1.
```

A. It returns the IDs of the candidates, sorted by ascending order, of those whose names are in the first table and have greater than 3 years of experience with a data mining course taken.
B. It returns the ordered IDs of the candidates of those whose names are in the first and not the second table and have greater than 3 years of experience with a data mining course taken.
C. It returns the IDs of the candidates, sorted by ascending order, of those whose names are in both tables and have greater than 3 years of experience with a data mining course taken.
Answer:

50. **Learning Objective: SQL Queries.** Refer to the scenario above. If the company slightly modifies the code above as a part of query optimization by removing all but the first and the last two lines and inserting `CROSS JOIN T2 ON T1.Name = T2.Name` after the new first line, does the resulting code perform the same operation as the original code above?
A. Yes B. No C. It depends on the inputted database.
Answer:

Questions 16-20 refer to the following scenario. We examine the likelihood of drawing candidate colorings for the pair-wise positioned Markov random fields using the Gibbs free energy. With a Markov random field over n nodes, we express a single coloring as a vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ representing the probability of candidate i $(i = 1, 2, \ldots, n)$ drawing color x_i given candidate i was only $(x_i = 1)$. A particular coloring was not yet drawn if x_i represents 0 votes. The k colors in the particular Markov field is drawn from $\{\text{red}, \text{blue}, \text{green}\}$ with drawing the most probable for each node in the Markov field. Refer to the included graph and answer the following questions.



Node	Node Probability
x_1	$\{1, 2\}$
x_2	$\{1, 2, 3\}$
x_3	$\{1, 2, 3\}$

16. (Learning Objective: Maximum Likelihood Estimation) When n the optimal $\mathbf{x}$ of k parameters $\{\theta_1, \theta_2, \ldots, \theta_k\}$ from the $\ldots$ the likelihood function that Maximizes (Minimizes) which will output the most negative $\{\theta_1, \ldots, \theta_k\}$?

   a. $\ldots$ (a) b. $\ldots$ (b) c. $\ldots$ (c) d. $\ldots$ (d)

   a. $P(\ldots)$ b. $P(\ldots)$ c. $P(\ldots)$ d. $P(\ldots)$

   Answer:

17. (Learning Objective: Convolution) Consider the neighborhood system N produced from the Markov model above. With $\mathbf{x}$ representing arbitrarily assigned $\mathbf{x}$. Which of the following is not necessary conditions that must hold for the output of applying "valid" convolution over the Markov network to produce a neighborhood system? Select the maximum number of possible options that apply.

$$N = \begin{bmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_1 \\ \mathbf{x}_2 & \ldots & \mathbf{x}_2 \\ \mathbf{x}_3 & \ldots & \mathbf{x}_3 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$$

   a. $\ldots$ b. $\ldots$ c. $\ldots$
   d. $\ldots$ e. No such combinations can exist.

   Answers:

18. (Learning Objective: Graph Theory) A new heuristic depth-first search algorithm is created that finds the first k such paths connected to a given neighbor of the node. In the problem, all k edges have an the most weighted k edges are selected, starting from $\{\ldots\}$ and including the selection of the next node. A researcher tries heuristic depth-first search based traversal algorithm with a new feature k to go from these assignments, say the next most node, as the Markov model above with nodes, and $\{\ldots\}$ paths to total. Which $\binom{n}{k}$ represents all combinations of valid traversals among the three nodes for a particular start node.

   Pros: **Cons:**

Appendix: Statistical Theorems, Formulae, and Data Structures

- Conditional Probability & Bayes' Theorem: $P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A) \cdot P(A)}{P(B)}$
- Expectation Properties: (1) $E(X+Y) = E(X) + E(Y)$, (2) $E(aX+bY) = aE(X) + bE(Y)$
- Sample Variance:

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

- Variance Properties: (1) $Var(X) = E(X^2) - (E(X))^2$, (2) $Var(aX+bY) = a^2Var(X) + b^2Var(Y)$
- Covariance Formula: $Cov(X,Y) = E[(X - \mu_X)(Y - \mu_Y)] = E(XY) - E(X)E(Y)$
- Correlation Formula: $Corr(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$ (if $Corr(X,Y) = 0$, r.v.'s X and Y are unrelated)
- Independence Rule: $f_{X,Y}(x,y) = f_X(x) f_Y(y)$ if r.v.'s X and Y are independent.
- Standardization of Random Variables: $X' = \frac{X - \mu_X}{\sigma_X}$ (transforms r.v. to have $\mu = 0$ and $\sigma = 1$)
- Conditional PDFs: $f_{Y|X}(y|x) = \frac{f_{X,Y}(x,y)}{f_X(x)}$
- Property of Valid Density Functions: $\int\int_{\mathbb{R} \times \mathbb{R}} f_{X,Y}(x,y)\, dy\, dx = 1$
- Binomial distribution (discrete): $f_X(x; n, \rho) = \binom{n}{x} \rho^x (1-\rho)^{n-x}$, $x \in 0, 1, ..., n$
- Geometric distribution (discrete): $f_X(x; \rho) = \rho(1-\rho)^{x-1}$, $x \in 1, 2, ..., n$ (r.v. X represents the number of failures until obtaining the first success)
- Poisson distribution (discrete): $f_X(x; \lambda) = \frac{e^{-\lambda} \lambda^x}{x!}$, $x \in 1, 2, ..., n$ (models the number of times an event occurs in a fixed length of time)
- Exponential Random Variable (continuous):

$$f_X(x; \lambda) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- Normal Random Variable (continuous): $f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma^2)}$, $x \in \mathbb{R}^1$

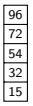


Figure 1: A Stack

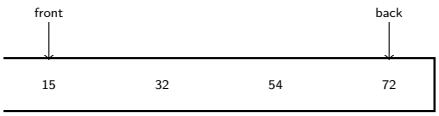


Figure 2: A Queue

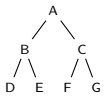


Figure 3: A Binary Tree

Section II (45 minutes)

(c) [Recall from above that the file is stored in Dropbox's S3 computer data. The networking systems running on Dropbox's S3 (BackBlaze Pro, max 40MB/s) will certainly not send packets to public (i.e. not VPN) Dropbox to overwrite. The result is that the number of the file's metadata, which Dropbox's system uses to indicate Dropbox systems exist in different sizes. An indirect block containing either a single, double, or triple indirect block tag. Each indirect block contains 12 block numbers. When the file is sent by the manager to Dropbox's S3 and after the file is stored in Dropbox's S3 table in sync the message. "Will agree to this". We have an issue. Dropbox's Data uses to compress the file, and further back, to retrieve the file's metadata. This file compression process is performed by a software. Compressing, which converts the original 50 MB file to a new 40 MB file. To retrieve this, the software modifies all data blocks pointed to by block pointers within direct and to indirect blocks. Assuming that each data block is 4kB, to update and to of a block size, the data blocks within clusters of the following blocks will be impact can be the software?

I. Direct Blocks only
II. Direct Blocks and Single Indirect Block
III. Single Indirect Block and Double Indirect Block
IV. Double Indirect Block only
V. Triple Indirect Block only

Answer: ☐

(d) You know that the company use Dropbox, and that the file being manipulated was a graph database. Graph and relational networks will You are a team of data warehouse that use conventional network networks with inputs as graphs within. Also the conventional function used. When you be used in the context of interrogation/warehousing to identify nodes with missing labels in a graph. The first step is to identify node is another is constructing an adjacency matrix to represent the connections of the graph.

In the space below, write down the adjacency matrix corresponding to the undirected graph in the right and containing labels connections.

Question. (d) (c) How does the Google search engine the possible library to the most modern of a business you use? This is through the interconnected mechanisms artificial intelligence was introduced.

You are a consultant of Google's algorithms and Mathematician team working in conjunction with data managing team. You are told to implement a language model — a probabilistic model that assigns probabilities to sentences. Given a different length of words as possible word sequence of the sentence, you could to determine which words the most complete sentences. Specifically, you are asked to predict the next few words of an existing 3-word sentence, as problems $w_1 w_2 w_3$. You are instructed that the probability of a word being chosen is independent of the three words of the sequence. Given below algorithm the algorithm would incredible speed of a possible to a recursive dynamic programming approach, and complete function. [show suggestions] [1] Solve:

Greedy **Recursive Dynamic Programming**

```
Complete the probability of word based words [1] over dictionary implementing the training
def get_word_prob_given_previous_words( dictionary ):
    pass
Assume a text is taught to constructing the the words the dictionary containing will return
    the probability that a word follows a given sequence of words in the text. For example,
    if the word 'cat' follows 'the' 4 times in the text, and 'the' appears 100 times in
    training set, then the probability the third is cat is 4/100.
def word_sequence_from_file( filename ):
    pass
```

(e) Show How did

Question #5. You have the following code written in an IPython Notebook to train a deep learning architecture defined in PyTorch with image data. You observe that after many epochs, the validation error is significantly higher than the training error. You want to reduce this gap by adding a layer to your network. Fill in blanks `(1)` and `(2)` below to achieve this effect, and blanks `(3)` to `(5)` to iterate through two batches of the training data and display all the images within those batches.

[1]:
```
import matplotlib.pyplot as plt
import torch
from torch import nn, optim
from torch.utils.data import DataLoader
from torchvision import models, transforms, datasets
import torch.nn.functional as F
from torchvision.transforms import ToTensor, Lambda, Compose
```

[2]:
```
class Network(nn.Module):
    def __init__(self): # Constructor
        super(Network, self).__init__()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(28*28, 512)
        self.fc2 = nn.Linear(512, 10)
        (1)_________________________

    def forward(self, x):
        x = x.view(x.size(0), -1)
        x = F.relu(self.fc1(x))
        (2)_________________________
        x = F.relu(self.fc2(x))
        x = F.softmax(x)
        return x
```

[3]:
```
model = Network()
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
transform = transforms.Compose([
    ToTensor(),
    transforms.RandomHorizontalFlip(),
    transforms.RandomVerticalFlip()
])
train_data = ...
train_loader = DataLoader(train_data, batch_size=10, shuffle=True)
```

[4]:
```
def train_model(model, device, train_loader, optimizer, epochs):
    ...

train_model(model, 'cuda', train_loader, optimizer, 5)
```

[5]:
```
dataiter = iter(train_loader)
images_batch1, labels_batch1 = (3)_________________
images_batch2, labels_batch2 = (4)_________________
```

[6]:
```
# batch 1
fig, axs = plt.subplots(1, 10, figsize=(28,28))
for i in range(10):
    axs[i].imshow((5)_________________)
# batch 2
...
```

STOP 2. If you finish before time is called, you may check over your work.

A Practical Guide to Artificial Intelligence and Data Analytics

Due to a limited number of practice facilities in the field of Data Science and Artificial Intelligence, I have written this book to deliver concepts in a simple and clear manner with practical applications from case studies and real-world scenario-based exercises to facilitate the learning process. I also believe that one effectively learns material when after learning the material, they are given an assessment to measure their aptitude. As a result, I have included a full-length assessment, titled *Data Science and Analytics Skills Assessment (DSSA)*, to create a checkpoint that allows one to maximize their study-skills. From this assessment, they can learn from their mistakes, and address holes in their understanding. And, the *Data Science and Analytics Skills Assessment* is specifically designed to pinpoint these holes, giving a score that accurately reflects their level of understanding.

ISBN 979-87-9395-048-0