notes.md

Introduction

About the exam

Recommended

- At least an Associate certification or equivalent experience
- 2 years of hands-on experience in AWS
- Some experience with general machine learning topics and algorithms

Exam:

- 180 minutes, ~65 questions
- Multiple choirce and multiple-resonse
- No partial credit
- No points for unanswered questions
- Scores between 100 and 1000 with min passing score 750
- Scaled scoring models used

Domains:

1. Data Engineering - 20%
2. Exploratory Data Analysis - 24%
3. Modeling - 36%
4. Maching Learning Implementation and Operations - 20%

AWS will try to trick you to expose gaps in your knowledge.

About Course

- Deep knowledge of Sagemaker, Glue, and Kinesis
- Focus on service capabilities and what problems they solve, don't memorize service limits, prices, version numbers.
- Study to become and Expert, don't study to pass the Exam
- Make it your own
- Review the Supplemental Material

Data Collection

- Machine Learning Cycle - Fetch -> Clean -> Prepare -> Train Model -> Evaluate Model -> Deploy to Production -> Monitor & Evaluate -> Fetch ...
- Goal
    - Understand the problem at hand.
    - Understand the parts of our input data

- Map our data into AWS

Data Collection Concepts

Before you begin, ask a few questions:

- What type of generalization are we seeking?
  - Are we trying to forecast a number? See if the customer will pick option A or B?
- Do we really need machine learning?
  - Could it just be done with if/then logic?
- How will my ML generalization be consumed?
  - Will we need to return results in real time, or just in some batch process?
  - Will others need to talk to this using an API?
- What do we have to work with?
  - What kind of data is out there/in house that we can use with our machine learning process?
- How can I tell if the generalization is working?

Traits of Good Data	Traits of bad data	Why
Large datasets	Small datasets (less than 100 rows).	Generally, more data means better model training
Precise attribute types, feature rich	Useless attributes, not needed for solving problem at hand.	Models need to train on importan features
Complete fields, no missing values	Missing values, null fields.	Modes can skewresults when data points are missing.
Values are consistent.	Inconsistent values	Models like clean and consistent data
Solid distribution of outcomes	Lots of positive outcome, few negative outcomes	Models cannot learn with skewed distribution of outcomes
Fair sampling	Biased sampling	Models will skew results with biased data

You should have at least 10 times as many data points as the total number of features.

No matter how you get your data, you are building a data repository. We must find a way to congregate the data into a single data repository.

General Data Terminology

Data is an integral part of machine learning. Understanding the terminology allows us to define and describe our data.

Datasets - The data we use in machine learning is usually defined as a dataset. Datasets are a collection of data.

- dataset = input data = training/testing data
- column = attribute = feature
- row = observation = sample = data point

Structured Data - has a defined schema and a schema is the information needed to interpret the data, including attribute names and their assigned data types.

Unstructured Data - has no defined schema or structural properties. Makes up majority of data collected.

Semi-structured Data - too unstructured for relational data, but has some organizational structure. Usually in the form of csv, json, or xml.

Database

- Traditional relational databases
- Transactional
- Strict defined schema

Data Warehouse

- Processing done on import (schema-on-write)
- Data is classified/stored with user in mind
- Ready to use with BI tools (query and analysis)

Data Lakes

- Processing done on export (schema-on-read)
- Many different sources and formats
- Raw data may not be ready for use

Machine Learning Data Terminology

Labeled Data - data where we already know what the target attribute is

Unlabeled Data - data that has been collected with no target attribute

Supervised learning is done with labeled data, unsupervised learning is done with unlabeled data.

Categorical features

- Values that are associated with a group
- Qualitative
- Discrete

Continuous features

- Values that are expressed as a measurable number
- Quantitative
- Infinite

Text Data (Corpus Data) - datasets collected from text. Used in Natural Language Processing (NLP), speech recognition, text to speech, and more.

Ground Truth - refers to factual data that has been observed or measured. This data has successfully been labeled and can be trusted as "truth" data.

Amazon SageMaker Ground Truth

- Tool that helps build ground truth datasets by allowing different types of tagging/labeling processes.
- Easily create labeled data.

Image Data - datasets with tagged images

Helpful image datasets:

- MNIST data - a collection of tagged handwritten characters to aid with handwriting analysis problems
- Image Net - a collection of tagged, searchable images to aid with image classification problems

Time Series Data - datasets that capture changes over time

Dataset Type	Example Cases	Format
Image Data	Facial recognition, action recognition, object detection, handwriting and character recognition	images, videos
Text Data	Reviews, news articles, messages, twitter and tweets, dialogs	text, csv
Sound Data	Speech, music, other sounds	mp3, text
Signal Data	Electrical signals, motion-tracking, chemical compounds	text
Physical Data	High-energy physics systems, astronomy, earth science	text
Biological Data	Human, animal, plants, microbes	text
Multi-variable Data	Financial, weather, census, transit, internet, games	csv, text

AWS Data Stores

Amazon Simple Storage Service (S3)

- Unlimited data storage that provides object based storage for any type of data
- Go to place for storing Machine Learning data
- Files can be from 0 bytes to 5 TB
- There is unlimited storage
- Files are stored into buckets (similar to folders)
- S3 is a universal namespace. That is, names must be unique globally
- Upload data through the console or cli/sdk

Relational Database Service (RDS) - SQL store for relational datasets

DynamoDB - NoSQL store for nonrelational datasets

Redshift - Data warehousing solution

Timestream - Allows BI acess to time series data

DocumentDB - somewhere to migrate mongodb data

AWS Migration Tools

Data Pipeline - Used to transfer data from other services to S3, also can be used as a transformation tool

DMS - Used to migrate data between different database platforms

AWS Glue - Fully Managed ETL (Extract, Transform, and Load) service

Datasource	Migration Tool	Why
PostgreSQL RDS instance with training data	AWS Data Pipeline	Specify SqlActivity query and places the output into S3
Unstructured log files in S3	AWS Glue	Create custom classifier and output results into S3
Clustered Redshift data	AWS Data Pipeline or AWS Glue	Use the unload command to return results of a query to CSV file in S3 or Create data catalog describing data and load it into S3

Datasource	Migration Tool	Why
On-premise MySQL instance with training data	AWS Database Migration Service	DMS can load data in CSV format onto S3

AWS Helper Tools

EMR - Distributed workloads over many EC2 systems. Hadoop cluster to process, transform, and analyze large amounts of data. Store petabytes of data in various platforms

Athena - Serverless SQL queries in S3 data

Redshift Spectrum

- Query S3 data
- Must have Redshift Cluster
- Made for existing customers

Athena

- Query S3 data
- No need for Redshift cluster
- New customers quickly want to query S3 data

Exam Tips

Before you Begin

- Understand that before we gather input data we must formulate the problem we are trying to solve
- Know how we can measure success and what your goals are
- Determine if Machine Learning is even necessary
- Understand what type of data is available to help solve problem.

Good Data

- Understand what makes up "good" data and why having good data is important
- Understanding what "good" and "bad" data looks like

Data Terminology

- Know how to identify columns/attributes and rows/observations within a dataset
- Know the difference in structured, semi-structured, and unstructured data
- Know the different types of data repositories (databases, data warehouses, data lakes)
- Understand the differences between labeled data and unlabeled data
- Be able to recognize categorical features and continuous features
- Know terms like corpus, ground truth, time series data, and image data

AWS Data Stores Tools

- Know the different AWS services where data can be stored
- Know what types of data is stored in different AWS services

AWS Migration Tools

- Know the Different AWS services we can use to migrate data

- Know when to use one migration tool over another

AWS Helper tools

- Know what EMR is and how we could use it as a migration tool
- Know what Amazon Athena is and how it differs from Redshift Spectrum.

Streaming Data Collection

Streaming Data Collection Concepts

Static Data - Data collected and stored

Free Dataset Resources

- Kaggle - large quantity of free datasets
- UCI - many OS datasets
- OpenData on AWS
- Google BigQuery

Streaming Data - data streamed in realtime

Kinesis tools used to handle streaming data

- Kinesis Data Streams
- Kinesis Data Firehose
- Kinesis Video Streams
- Kinesis Data Analytics

Kinesis Data Streams

Gets data from producers, the things that produce the data we want in AWS

Can use Kinesis Streams to get the streaming data into AWS using shards (container that holds data we want in aws)

Then use consumers to process that data and store in some storage location

Shard Components

- Partition Key - unique for each shard
- Sequence - each time we make a request, it creates a sequence associated with a shard
- Data

Shard Notes:

- Each shard consists of a sequence of data records. These can be ingested at 1000 records per second
- Default limit of 500 shards, but you can request to unlimited shards.
- A data record is the unit of data captured
    - sequence number
    - partition key
    - data blob (your payload, up to 1 MB)
- Transient Data Store - retention period for the data records are 24 hours to 7 days.

Interacting with Kinesis Data Streams

1. Kinesis Producer Library (KPL) - Easy to use library that allows you to write to a Kinesis Data Stream
2. Kinesis Client Libray (KCL) - Integrated directly with KPL for consumer applications to consume and process data from Kinesis Data Stream
3. Kinesis API (AWS SDK) - Used for low level API operations to send records to a Kinesis Data Stream

Kinesis Producer Library (KPL)

- Provides a layer of abstraction specifically for ingesting data
- Automatic and conrfigurable retry mechanism
- Additional processing delay can occur for higher packing efficiencies and better performance
- Java wrapper

Kinesis API

- Low-level API calls (PutRecords and GetRecords)
- Stream creations, resharding, and putting and getting records are manually handled
- No delays in processing
- Any AWS SDK

When should you use Kinesis Data Streams?

- Needs to be processed by consumers
- Real time analytics
- Feed into other services in real time
- Some action needs to occur on your data
- Storing data is optional
- Data retention is important

Use Cases

- Process and evaluate logs immediately
  - Example: Analyze system and application logs continuously and process within seconds.
- Real-time data analytics
  - Example: Run real-time analytics on click stream data and process it within seconds.

Kinesis Data Firehose

Data comes from Data Producers, can be preprocessed using AWS Lambda, and is sent to some storage solution (Redshift, S3, ...)

Used to stream data directly to a final data storage location / destination

When should you use Kinesis Data Firehose?

- Easily collect streaming data
- Processing is optional
- Final destination is S3 (or other data store)
- Data retention is not important

Use Cases

- Stream and store data from devices
  - Example: Capturing important data from IoT devices, embedded systems, consumer applications and storing it into a data lake
- Create ETL jobs on streaming data

- Example: Running ETL jobs on streaming data before data is stored into a data warehousing solution

Kinesis Video Streams

Allows us to stream videos into the AWS cloud/images/audio files in real time.

Producers like webcams, microphones, live video feeds, etc. Consumers process data in fragments and frames from KVS to view, process and analyze it Then store in S3 if desired

When should you use Kinesis Video Streams?

- Needs to process real-time streaming video data (audio, images, radar)
- Batch-process and store streaming video
- Feed streaming data into other AWS services

Kinesis Data Analytics

Allows you to continuously read and process streaming data in real time using SQL queries. Gets input from Kinesis Data Streams/Kinesis Data Firehose, run real time SQL queries, and output the results in Redshift, S3, visualization/BI tools

When should you use Kinesis Data Analytics?

- Run SQL queries on streaming data
- Construct applications that provide insight on your data
- Create metrics, dashboads, monitoring, notifications, and alarms
- Output query results into S3 (or other AWS datasources)

Use Cases

- Responsive real-time analytics
  - Example: Send real-time alarms or notifications when certain metrics reach predefined threshold
- Stream ETL jobs
  - Example: Stream raw sensor data, then clean, enrich, organize, and transform it before it lands into data warehouse or data lake

Task at hand	Which Kinesis service to use?	Why?
Need to stream Apache log files directly from (100) EC2 instances and store them into Redshift	Kinesis Firehose	Firehose is for easily streaming data directly to a final destination. First the data is loaded into S3, then copied into Redshift
Need to stream live video coverage of a sporting event to distribute to customers in near real-time	Kinesis Video Streams	Kinesis Video Streams processes real-time streaming video data (audio, images, radar) and can be fed into other AWS services.
Need to transform real-time streaming data and immediately feed into a custom ML application	Kinesis (Data) Streams	Kinesis Streams allows for streaming hug amounts of data, process/transform it, and then store it or feed into custom applications or other AWS services.
Need to query real-time data, create metric graphs, and store output into S3	Kinesis Analytics	Kinesis Analytics gives you the ability to run SQL queries on streaming data, then store or feed the output into other AWS services

Exam Tips

Loading Data into AWS

- Understand how to get data from public or in house data sets and load it into AWS.
- Know the different ways to upload into S3 by using the console, the S3 API, or the AWS cli

The Kinesis Family

- Know what each service is and how it processes/handles streaming data
- Know what shards are, what a data record is, and the retention period for a shard
- Know the difference in the KPL, KCL, and Kinesis API
- For a given scenario, know which streaming Kinesis service to use.

Data Preparation

Data Preparation Concepts

Data Preparation

- The process of transforming a dataset using different techniques to prepare it for model training and testing
- Changing our dataset so it is ready for machine learning

Categorical Encoding - Converting categorical values into numeric values using mapping and one-hot techniques

Feature Engineering - Transforming features so they are ready for ML algorithms. Ensures the relevant features are used for the problem at hand.

Handling Missing Values - Removing incomplete, incorrect formatted, irrelevant or duplicated data

Options for Data Preparation

- SageMaker & Jupyter Notebooks
- ETL jobs in AWS Glue

Categorical Encoding

Categorical Encoding

- The process of manipulating categorical variables when ML algorithms expect numerical values as inputs
- Changing category values in our dataset to numbers

categorical variable = categorical feature = discrete feature

When to encode?

Problem	Algorithm	Encoding
Predicting the price of a home	Linear Regression	Encoding necessary
Determine whether given text is about sports or not	Naive Bayes	Encoding not necessary
Detecting malignancy in radiology images	Convolutional Neural Network	Encoding necessary

Categorical Encoding Examples:

- Color: {green, purple, blue}, multicategorical
- Evil: {true, false}, binary categorical

- Size: {L > M > S}, ordinal

Nominal - order does not matter Ordinal - order does matter

Binary Categorical - can map to 0 & 1

Multicategorical Ordinal - can map to integers (e.g. S to 1, M to 5, L to 10)

Multicategorical Nominal - shouldn't map to integers, implies order that's not there

One-hot Encoding

- Transforms nominal categorical features and creates new binary columns for each observation
- Adding columns to your dataset of 1's and 0's
- One-hot encoding is not always a good choice when there are many, many categories
- Using techniques like grouping by similarity could create fewer overall categories before encoding
- Mapping rare values to "other" can help reduce overall number of new columns created

Categorical Encoding Summary

1. In general, categorical encoding is used when the ML algorithm cannot support categorical data
2. We must find a way to turn text attributes into numeric attributes within our datasets.
3. There is no "golden rule" on how to encode your categories (or transform your data in general)
4. There are many different approaches and each approach can have a different impact on the outcome of your analysis

Text Feature Engineering

Feature Engineering - Transforming attributes within our data to make them more useful within our model for the problem at hand. Feature engineering is often compared to an art

Text Feature Engineering

- Transforming text within our data so Machine Learning algorithms can better analyze it
- Splitting text into bite size pieces

Example:

> Raw Text: {"123 Main Street, Seattle, WA 98101"}

With basic processing, can get something more useful:

Address	City	State	Zip
123 Main Street	Seattle	WA	98101

Bag-of-Words

Tokenizes raw text and creates a statitical representation of the text

Breaks up text by white space into single words

Example:

> Raw Text: {"he is a jedi and he will save us"} -> Bag-of-Words -> {"he", "is", "a", "jedi", "and", "he", "will", "save", "us"}

Becomes:

Id	word	count

Id	word	count
1	he	2
2	is	1
3	a	1
4	jedi	1
5	and	1
6	will	1
7	save	1
8	us	1

N-Gram

An extension of Bag-of-Words which produces groups of words of *n* size

Breaks up text by white space into groups of words

Example:

> Raw Text: {"he is a jedi and he will save us"} -> N-gram, size = 1 -> {"he", "is", "a", "jedi", "and", "he", "will", "save", "us"}

Same as Bag-of-Words

> Raw Text: {"he is a jedi and he will save us"} -> N-gram, size = 2 -> {"he is", "is a", "a jedi", "jedi and", "and he", "he will", "will save", "save us", "he", "is", "a", "jedi", "and", "he", "will", "save", "us"}

Uses a sliding window of size 2, once it finds all the pairs, also produces any n's less than its size.

unigram = 1 word tokens

bigram = 2 word tokens

trigram = 3 word tokens

...

Orthogonal Sparse Bigram (OSB)

Creates groups of words of size *n* and outputs every pair of words that includes the first word

Creates groups of words that always include the first word

Meaning:

- Orthogonal - when two things are independent of each other
- Sparse - Scattered or thinly distributed
- Bigram - 2-gram or two words

OSB is not "better" or "worse" than N-Gram, it's just another technique to use...

Example:

```
Raw Text: {"he is a jedi and he will save us"} -> OSB, size = 4 ->
  {"he_is", "he__a", "he___jedi"}
  {"is_a", "is__jedi", "is___and"}
```

```
{"a_jedi", "a__and", "a___he"}
{"jedi_and", "jedi__he", "jedi___will"}
{"and_he", "and__will", "and___save"}
{"he_will", "he__save", "he___us"}
{"will_save", "will__us"}
```

Always keeps the first word, then uses underscores where the other words in the token are

Term Frequency - Inverse Document Frequency (tf-idf)

Represents how important a word or words are to a given set of text by providing appropriate weights to terms that are common and less common in the text

Shows us the popularity of a word or words in text data by making common words like "the" or "and" less important

Meaning:

- Term Frequency - How frequent does a word appear
- Document Frequency - Number of documents in which terms occur
- Inverse - Makes common words less meaningful

Example:

Document 1:

word	count
the	3
force	1
Luke	1
Skywalker	1
a	2

Document 2:

word	count
the	2
jedi	1
a	1
empire	1

Since the tokens "the" and "a" showed up in BOTH documents many times, these are deemed as less important

Vectorized tf-idf: (number of documents, number of unique n-grams)

Example: (2,7)

2 - number of documents

7 - number of unique n-grams

Use Cases

Problem	Transformation	Why
Matching phrases in spam emails	N-Gram	Easier to compare whole phrases like "click here now" or "you're a winner"
Determining the subject matter of multiple PDFs	Tf-idf & Orthogonal Spare Bigram	Filter less important words in PDFs. Then find common word combinations repeated throughout PDFs.

Simple Transformations

Remove Punctuation

- Sometimes removing punctuation is a good idea if we do not need them.

Example:

> Raw Text: {"Help me, Obi-Wan Kenobi. You're my only hope."} -> remove punctuation -> {"Help", "me" "Obi-Wan", "Kenobi", "You're", "my", "only", "hope"}

Lowercase Transformation

- Using lowercase transformation can help standardize raw text

Example:

> Raw Text: {"I Find Your Lack of Faith Disturbing"} -> lowercase -> {"i find your lack of faith disturbing"}

Cartesian Product Transformation

Creates a new feature from the combination of two or more text or categorical values

Combining sets of words together

Meaning:

- Cartesian - Rene Descartes, created Cartesian Coordinate System
- Product - multiplication

We are multiplying a set of words by another set of words to create a new feature

Example:

ID	textbook	binding
1	Python Data Science Handbook	Softcover
2	Visualization Analysis & Design	Hardcover
3	Machine Learning Algorithms	Softcover

Remove punctuation -> cartesian product (textbook, binding)

Returns:

Id	cartesian_product
1	{"Python_Softcover", "Data_Softcover", "Science_Softcover", "Handbook_Softcover"}
2	{"Visualization_Hardcover", "Analysis_Hardcover", "Design_Hardcover"}
3	{"Machine_Softcover", "Learning_Softcover", "Algorithms_Softcover"

Feature Engineering Dates

Date Formats:

- 2013-02-08T09
- 6 Mar 17 21:22 UT
- Mon 06 Mar 2017 21:22:23 z
- 09/28/2019

Extracted Information:

- Was it a weekend? Was it a week day?
- Was the date the end of a quarter?
- What was the season?
- Was the day a holiday?
- Was it during business hours?
- Was the World Cup taking place on this date?

Example:

Date
2015-06-17
2015-04-24
2015-02-12
2015-12-16
2015-03-14

date feature engineering ->

is_weekend	day_of_week	month	year
0	2	6	2015
0	4	4	2015
0	3	2	2015
0	2	12	2015
1	5	3	2015

Text Feature Engineering Summary

Technique	Function
N-Gram	Splits text by whitespace with window size *n*
Orthogonal Sparse Bigram (OSB)	Splits text by keeping first word and uses delimiter with remaining whitespaces between second word with window size *n*
Term Frequency - Inverse Document Frequency (tf-idf)	Helps us determine how important words are within multiple documents
Removing Punctuation	Removes punctuation from text

Technique	Function
Lowercase	Lowercase all the words in the text
Cartesian Product	Combines words together to create new feature
Feature Engineering Dates	Extracts information from dates and creates new features

Numeric Feature Engineering

Transforming numeric values within our data so Machine Learning algorithms can better analyze hem

Changing numeric values in our datasets so they are easier to work with

Common Techniques:

- Feature Scaling
  - Changes numeric values so all values are on the same scale.
    - Normalization
    - Standardization
- Binning
  - Changes numeric values into groups or buckets of similar values
    - Quantile Binning aims to assign the same number of features to each bin

Feature Scaling

scaling = feature scaling = normalization

Normalization scales numbers between 0 and 1

Outliers can throw off normalization!

Standardization - uses z-scores to smooth out values

Scaling Summary:

- Scaling features is required for many algorithms like linear/non-linear regression, clustering, neural networks, and more. Scaling features depends on the algorithm you use
- Normalization rescales values from 0 to 1 but doesn't handle outliers very well
- Standardization rescales values by making the values of each feature in the data have zero mean and is much less affected by outliers
- When you're done, you can always scale back the data to the original representation

Binning

Categorizes numerical data into more useful groups, e.g. ages into "30s and below", "30s-50s", and "50 and above"

We can end up with irregular bins which are not uniform

Can apply Quantile Binning to fix

Meaning:

- Quantile - equal parts
- Binning - grouping

Grouping things together in equal parts.

From there, apply categorical data to each bin and use other techniques to find correlation between our predicted value and our target attribute

age
24
45
18
32
40
76

Quantile Binning ->

age_bin
Youth
Young Adult
Youth
Young Adult
Adult
Adult

one-hot encoding ->

Youth	Young Adult	Adult
1	0	0
0	1	0
1	0	0
0	1	0
0	0	1
0	0	1

Quantile Binning Summary:

- Binning is used to group together values to reduce the effects of minor observation errors
- Quantile binning bins values into equal numbers of bins
- Optimum number of bins depends on the characteristics of the variables and its relationship to the target. This is best determined through experimentation

Numeric Feature Engineering Summary

Technique	Function
Normalization	From 0 to 1. 0 - minimum value, 1 - maximum value.
Standardization	0 is the average. Value is the z-score.

Technique	Function
Quantile Binning	Creates equal number of bins.

Other Feature Engineering

Image Feature Engineering

- Extracting useful information from images before using them with ML algorithms.
- Transforming images to find out useful information

Audio Feature Engineering

- Extracting useful information from sounds and audio before using them with ML algorithms
- Transforming audio data into something useful

Dataset Formats:

- File
  - Loads all of the data from S3 directly onto the training instance volumes
  - CSV
  - JSON
  - Parquet
  - Image files (.png or .jpg)
- Pipe
  - Your datasets are streamed directly from Amazon S3
  - recordIO-protobuf (creates tensor)

Feature Engineering is like a layered cake. Usually there are **multiple** layers of transformations done to properly **prepare** your **data**.

Handling Missing Values

Missing data can be represented in many different ways (*null, NaN, NA, None, etc.*) and handling missing values is an important data preparation step.

Why are values missing in the first place?

- Missing at Random (MAR)
  - Missing at random means that the propensity for a data point to be missing is not related to the missing data, but it is related to some of the observed data
- Missing Completely at Random (MCAR)
  - The fact that a certain value is missing has nothing to do with its hypothetical value and with the values of other variables
- Missing not at Random (MNAR)
  - Two possible reasons are that the missing value depends on the hypothetical value or missing values is dependent on some other variable's value

How to handle missing values:

Technique	Why this works	Ease of Use
Supervised learning	Predicts missing values based on the values of other values	Most difficult, can yield best results
Mean	The average value	Quick and easy, results can vary
Median	Orders values then chooses value in the middle	Quick and easy, results can vary

Technique	Why this works	Ease of Use
Mode	Most common value	Quick and easy, results can vary
Dropping rows	Removes missing values	Easiest but can dramatically change datasets

Replacing data is known as **imputation**

Feature Selection

Feature Selection

- Selecting the most relevant features from your data to prevent over-complicating the analysis, resolving potential inaccuracies, and removes irrelevant features or repeated information
- Deciding what to keep and what to get rid of

Feature selection is an **intuitive step** humans take to reduce the number of features.

Principle Component Analysis (PCA)

- An unsupervised learning algorithm that reduces the number of features while still retaining as much information as possible
- Reduces the total number of features in a dataset

Feature Selection Use Cases:

Problem	Technique	Why
Data is too large due to the large number of features	Principle Component Analysis (PCA)	Algorithm that reduces total number of features
Useless features that do not help solve ML problem	Feature Selection	Remove features that do not help solve the problem

Helper Tools

AWS Glue

- Setup a crawler that crawls your input datasource (*S3, DynamoDB, RDS, Redshift, Database on EC2*), determines the schema/structure of your dataset
- Crawler creates a Data Catalog, the metadata, datatypes, important data about dataset
- Then set up jobs in Scala or Python to transform the dataset
- Then output transformed data into data source (*Athena, EMR, S3, Redshift*) and use later for the ML process
- Job Types
  - Spark - default type, fully managed cluster of Apache Spark servers that Glue runs in the background and allows us to run transformation code on
    - ETL language - PySpark or scala, used to tranform data
    - How we want the code to be generated
      - AWS generate the script for us
      - Provide with a script of our own
      - Start a brand new script from scratch
  - Python shell
    - Traditional python scripts to run on your datasets
- Also allows you to use Zeppelin / Jupyter notebooks to do ad-hoc/simple transformations

SageMaker

- Allows you to create Jupyter notebooks that are directly integrated with the SageMaker service

Elastic Map Reduce (EMR)

- Entire data preparation ETL could be done on the EMR ecosystem.
- A fully managed hadoor cluster that runs on multiple EC2 instances
- Can pick and choose different frameworks that you want to use within the cluster
- Great at scaling petabytes worth of data over distributed systems
- Notable Tools
  - Apache Spark - ETL and Machine Learning Library
  - Presto - SQL Query Engine
  - Mahout - Machine Learning Framework
  - Hive - ETL Service
  - Jupyter Notebooks - Code Sharing
  - TensorFlow - Machine Learning Framework
  - Hadoop Distributed File System (HDFS) - Persistant Datastore
  - MXNet - Machine Learning Framework
- Given data stored in our EMR cluster, how do we use it within SageMaker?
  - Can use Apache Spark to integrate directly within SageMaker
- How do we perform the least amount of effort for ETL jobs?
  - Using AWS Glue, because it's fully managed and we don't have to manage the cluster ourselves in EMR

Athena

- Run SQL Queries on S3 data
- Managed by AWS, as long as you have data catalog set up, you can query it with Athena

Data Pipeline

- Process and move data between different AWS Compute services
- Moving data from DynamoDB, RDS, Redshift, through Data Pipeline, doing ETL jobs with EC2 instances/EMR, landing output dataset in target data source (*Athena, EMR, S3, Redshift*)
- Can choose from list of built in templates to migrate from one service to another

Which service to use?

Datasource	Data Preparation Tool	Why
S3, Redshift, RDS, DynamoDB, On Premise DB	AWS Glue	Use Python or Scala to transform data and output data into S3
S3	Athena	Query data and output results into S3
EMR	PySpark/Hive in EMR	Transform petabytes of distributed data and output data into S3
RDS, EMR, DynamoDB, Redshift	Data Pipeline	Setup EC2 instances to transform data and output data into S3, if you wanted to use some language other than Python or Scala

Exam Tips

Data Preparation

- Know what data preparation is and why it is important in Machine Learning
- Understand the different techniques used for preparing data

Categorical Encoding

- Know why categorical encoding is used for certain ML algorithms
- Understand the difference between ordinal and nominal categorical features
- Understand that categorical data is qualitative and continuous data is quantitative
- Know what one-hot encoding is and where to use it

Numeric Feature Engineering

- Know what numeric feature engineering is and why it is important
- Know different techniques used for feature engineering numeric data
- Know the different types of feature scalin and when they should be used
    - Normalization
    - Standardization
- Know what binning is and when it should be used

Text Feature Engineering

- Know what text feature engineering is and why it is important
- Know different techniques used for feature engineering text data
    - N-Gram
    - Orthogonal Sparse Bigram (OSB)
    - Term Frequency-Inverse Document Frequency (tf-idf)
    - Removing punctuation
    - Lowercase transformation
    - Cartesian product
- Understanding why feature engineering dates is important
- Know the questions we can answer when dates are transformed

Other Feature Engineering

- Know other types of feature engineering covered

Handling Missing Values

- Know why handling missing values is an important step in data preparation
- Know the different techniques used for handling missing values
- Understand implications of dropping rows
- Understand what data imputation is

Feature Selection

- Know what feature selection is and why it is important
- Understand the difference in feature selection and Principle Component Analysis (PCA)

Data Preparation Tools

- Know the different AWS services that allow you to transform data
- Know what a Data Catalog, Crawlers, and Jobs are in AWS Glue
- Be able to identify the different AWS servics and when to use one transformation over another

Data Analysis and Visualization

Data Analysis and Visualization Concepts

Seeing your data

- Relationships
  - Do we want to find important relationships within our data? Are there any trends or outliers?
- Comparisons
  - Are we comparing different values within our data?
- Distributions
  - Do we want to know more about the distributions of our data? Are there any outliers?
- Compositions
  - Do we want to know what makes up our data? What are the different parts of our data as a whole?
- Ways to see your data
  - Developer Tools (matplotlib, R)
  - Business Intelligence Tools (BI Tools) (Tableau, Amazon QuickSight)

Amazon QuickSight

- Business intelligence tool (BI tool) that makes it easy to create visualizations from your data
- Create awesome visualizations in the AWS console

Relationships

Visualizing relationships in your data can provide a good general overview, show distribution, and correlation between attributes. Visualizing relationships can also help find outliers and extreme values.

Scatter Plots - Also known as scatter charts. These graphs plot points along the x and y axis for two values.

Bubble Plots - Also known as bubble charts. These graphs plot points along the x and y axis for three values. Bubble size is the third value measured.

Use Cases

- Scatter Plots
  - Example: Is there any relationship between the size of a home and the price?
- Bubble Plots
  - Example: Are there any relationships between the size of a home, the age of the home, and the price?

Comparisons

Visualizing comparisons in your data can provide a static snapshot of how different variables compare and show how different variables change over time.

Bar Charts - graphs that use line (bars) to mark single variable values. Provides a way to lookup and compare values

Line Charts - graphs that use lines to show one or more variables changing over time

Use Cases

- Bar Charts
  - Example: Comparing the number of star ratings for a mobile application in the app store
- Line Charts

- Example: Plotting the price of Bitcoin over the past 3 years

Distributions

Visualizing distributions in your data can show how your data is grouped or clustered over certain intervals.o

Histograms - put values into buckets or bins and determine a measurement (amount, frequency, duration, density, etc.)

Box Plots - show a wealth of distribution information. You can see things like lowest and highest values, outliers, and where most of the values fall

Scatter Plots - Also known as scatter charts. These graphs plot points along the x and y axis for two values. Can show clustering and distribution of your data

Use Cases

- Histograms & Box Plots
  - Example: Showing the distribution of test scores for a given exam
- Scatter Charts
  - Example: Showing the return of investment (ROI) for the amount of money spent and the total time invested

Compositions

Visualizing composition of your data show the various elements and what your data is made of

Pie Charts - show how various values compare as a whole, share of the total

Stacked Area Charts - show the measurement of various items over longer periods of time

Stacked Column Charts - Also known as stacked bar charts. These graphs show the quantity of various items over shorter periods of time.

Use Cases

- Pie Charts
  - Showing the sales figures for each region
- Stacked Area Charts
  - Example: Showing the number of products sold by different departments on a weekly basis
- Stacked Bar Charts
  - Example: Showing the quarterly revenue totals for each region

Choosing a Visualization

Picking the right graph, chart, or visualization just depends on what you want to see

What do we want to show?

- Relationship
  - 2 variables - Scatter Plot
  - 3 variables - Bubble Chart
- Comparison
  - Value look ups - Bar Charts
  - Change over time - Line Chart
- Distribution
  - Single distribution - Histogram
  - Multi distribution - Box Plots / Scatter Plots

- Composition
  - Changing over time - Stacked Bar Chart / Stacked Area Chart
  - Static - Pie Chart

Heatmaps are graphs that represent values as color. As the values change, the color representation of the data changes too.

Exam Tips

Data Analysis and Visualization

- Know what data analysis and visualization is and why it is important
- Understand what Amazon QuickSight is and how it can be used.
- Be able to recognize different types of visualizations and what each visualization represents
  - Relationships
  - Comparisons
  - Distribution
  - Composition
- Know what heatmaps are and what they can represent

Modeling

Modeling Concepts

What is a model?

- Taking a problem or challenge as described by lots of data, adding a machine learning algorithm and through computation, trying to figure out a mathematical formula that can accurately generalize about that problem.

Questions to ask for developing a good model

- What type of generalization are we seeking?
  - Do I need to forecast a number? Decide whether a customer is more likely to choose Option A or Option B? Detect a quality defect in a machined part?
- Do we really need machine learning?
  - Can simple heuristics handle the job just as well? Can I just program some IF...THEN logic? Will a linear regression formula or a look-up function fulfill the needs?
- How will my ML generalizations be consumed?
  - Do I need to return real-time results or can I process the inferences in batch? Will consumers be applications via API call or some other systems which will perform additional processing on the data?
- What do we have to work with?
  - What sort of data accurately and fully captures the inputs and outputs of the target generalization? Do I have enough data? Do I have too much?
- How can I tell if the generalization is working?
  - What method can I use to test accuracy and effectiveness? Should my model have a higher sensitivity to false positives or false negatives? How about Accuracy, Recall, and Precision?

	Supervised Learning	**Unsupervised Learning**	**Reinforcement Learning**
Discrete	Classification	Clustering	Simulation-based Optimization
Continuous	Regression	Reduction of Dimentionality	Autonomous Devices

Choosing the Right Approach:

Problem	Approach	Why
Detect whether a financial transaction is fraud	Binary Classification	Only two possible outcomes: Fraud or Not Fraud
Predict the rate of deceleration when brakes are applied	Heuristic Approach (No ML Needed!)	Well-known formulas involving speed, inertia and friction to predict this
Determine the most efficient path of surface travel for a robotic lunar rover	Simulation-based Reinforcement Learning	Must figure out the optimal path through trial, error and improvement
Determine the breed of dog in a photograph	Multi-Class Classification	Which dog breed is most associated with the picture among many breeds?

Sometimes you have to combine algorithms to solve the problem

Example: What is the estimated basket size of shoppers who respond to our email promotion?

Steps:

- Remove outliers (Random Cut Forest) ->
  - Identify relevant attributes (PCA) ->
    - Cluster into groups (K-Means) ->
      - Predict basket size (Linear Learner)

Confusion Matrix:

Columns are actual outcome

Rows are predicted outcome

	TRUE	FALSE
TRUE	I predicted correctly!	I was wrong. (False Positive)
FALSE	I was wrong. (False Negative)	I predicted correctly!

Problem: Is a given financial transaction fraudulent?

	Fraud	Not Fraud
Fraud	Happy Bank. Happy Customer. (No Money Loss)	Happy Bank. Angry Customer. (No Money Loss)
Not Fraud	Angry Bank. Angry Customer. (Money Lost!)	Happy Bank. Happy Customer. (No Money Loss)

Evaluation Approach

- The Bank is likely ok with more false positives than false negatives as it further reduces their exposure to fraud.
- We'd closely look at Recall

Problem: Is this email spam?

	Spam	Not Spam
Spam	Spam is blocked.	Legitimate emails are blocked.
Not Spam	Spam gets through.	Legitimate emails get through.

Evaluation Approach:

- It's better to let spam through than to block legitimate emails.
- Set the evaluation approach to be more error on the side of caution to let spam through

- We'd watch the Precision of the model closely

Data Preparation

We want **Generalization** not **Memorization**

Use most data to train, but reserve some data to see if the model has really learned to generalize and not just repeating what we've already shown it.

Normally we break off 20-30% of the data as testing data, train the model on the remaining training data, and evaluate using the testing data.

Testing data should be similar in makeup to the training data, otherwise we could have a high error rate.

Instead we want to:

- Randomize the training data
- Then Split into training set and test set
- Then train the model on the training data
- Then evaluate the model using the test data

That doesn't work for every dataset though. For timeseries data, we would be better served by a Sequential Split strategy, using the beginning of the data to train, and the end to test.

Beware of situations where the model and evaluation set are too dissimilar by way of descriptive statistics to be useful. This can happen when data is sorted by one of the columns in the dataset then split sequentially.

We can evaluate a dataset for this problem using K-Fold Cross-Validation: using K different training/test set splits and evaluating the error rate of each respectively.

- If the error rates of each different round (fold) is approximately equal to each other, the data was well randomized.
- If one of the rounds has a significantly higher error rate, maybe our data isn't as random as we thought it was.

SageMaker Modeling

SageMaker has 4 basic areas:

- Ground Truth - Set up and manage labeling jobs for training datasets using active learning and human labeling
- Notebook - Access a managed Jupyter Notebook environment
- Training - Train and tune models
- Inference - Package and deploy your machine learning models at scale

Options for Model Creation:

- SageMaker Console
- Jupyter Notebooks
- SageMaker SDK
- Apache Spark

Modeling Process

- Training Dataset and Test Dataset go into S3
- Model reads from S3

Several data formats are supported:

- Check the documentation for the respective algorithm for recommendations

ContentType	Recommended SplitType
application/x-recordio-protobuf	RecordIO
text/csv	Line
application/jsonlines	Line
application/json	None
application/x-image	None
image/*	None

Accept	Recommended AssembleWith
application/x-recordio-protobuf	None
application/json	None
application/jsonlines	Line

Most SageMaker algorithms accept CSV

- The Target Value should be in the first column with no header. Be sure the metadata Content-Type is "text/csv" in S3

For Unsupervised algorithms we specify the absence of labels

- The Target Value should be in the first column with no header. Notice we specify the label size in the metadata value.

For optimal performance, the optimized protobuf recordIO format is recommended.

- Using this format, we can take advantage of Pipe mode
- The advantage of recordIO and Pipe mode is that the data can be streamed from S3 to the learning instance requiring less EBS space and faster start-up.

CreateTrainingJob API

- Two Options
  - High-Level Python library provided by Amazon SageMaker
  - Use the SDK for Python
- Doing the same things
  - Specify the training algorithm
  - Supply algorithm-specific hyperparameters
  - Specify the input and output configuration

Difference between parameter and hyperparameter

- Hyperparameters - Values set **before** the learning process
- Parameters - Values **derived via** the learning process

SageMaker Training

When you call the SageMaker library, in the background, SageMaker uses an Elastic Container Repository, which has Training Images and Inference Images for the various ML Algorithms.

Training is very math intensive, but inference typically isn't, so the training images tend to be much higher powered.

CreateTrainingJob API Call - References the Training Images

CreateModel API Call - References the Inference Images

Training

These image repositories are in different regions, select one near you.

Algorithm Name	Channel Name	Training Image and Inference Image Registry Path	Training Input Mode	File Type	Instance Class
BlazingText	train	`<ecr_path>`/blazingtext:`<tag>`	File or Pipe	Text file (one sentence per line with space-separated tokens)	GPU (single instance only) or CPU
DeepAR Forecasting	traing and (optionally) test)	`<ecr_path>`/forecasting-deepar:`<tag>`	File	JSON Lines or Parquet	GPU or CPU
...					

Training Image Selection:

- Training Images are named differently for which algorithm you want to use.
- Also have a channel name to keep straight whether you're training, testing validating, etc.
- Training Image and Inference Image Registry Path differ from algorithm to algorithm. If using the SageMager Python Library, it will automatically know most paths.
- Note the Tag attribute. It is a form of versioning for the repository.
  - Use **:1** version flag to ensure the stable version
  - Use **:latest** version tag for up-to-date but potentially not backward compatible
  - Use **:1** for production purposes
- Training Input File Mode = type of input data accepted. Recall that Pipe has better performance.
- File Type = Format of data accepted by algorithm. Recall the recordIO protobuf has best performance
- Instance Class = type of instance required for algorithm
  - Note that some are GPU only or CPU only
  - Example: XGBoost implements an open source algorithm optimized for CPUs

Once we've selected our image, SageMaker will spin up our image in Elastic Container Service, access the data from S3, and output results to S3

You can also specify your own Dockerfile, which points to your own repository, and train the model with the CreateTrainingJob API call with ECR path to your image

- `nvidia-docker` is supported as well for GPU-optimized images.

When you want to deploy your model:

- Pulls inference images from Elastic Container Repository
- Hosts your model in Elastic Container Service
- Uses S3 as the source of data
- We could control access to that model using API Gateway for example

In the training process, there is a log of information logged that could be useful:

- Arguments provided
- Errors during training
- Algorithm accuracy statistics

- Timing of the algorithm
- Can view in CloudWatch or SageMaker console
- Common Errors:
  - Error in specifying a hyperparameter such as an extra one.
  - Invalid value for a hyperparameter
  - Incorrect protobuf file format

Training with Apache Spark:

- Can use the SageMaker Spark library to convert Spark DataFrames into protobuf, then send it out to S3, then access from the container as usual
- Useful if you already have a large investment in Spark and want to use that data in a training job/inferences

Exam Tips

Model Design

- Selecting a model that is a good fit for the objective
- Choosing the proper ML approach for your objective (regression, binary classification, etc.)
- Choose proper evaluation strategies for your model
- Steps for training a model

Data Preparation

- Understand concepts of Training Data and Testing Data
- Identify potential biases introduced in an insufficient split strategy
- Know when to use sequential splits vs randomized splits and what additional measures could be used to increase training data value

Model Training

- Multiple options for training
  - SageMaker Console
  - Apache Spark
  - Custom Code via SDK
  - Jupyter Notebook
- Be familiar with the default data types SageMaker algorithms support and the recommended format for best performance
- Know the difference between a Hyperparameter and a Parameter
- Understand the repository and container image concept for SageMaker training
- Understand the process if you wish to provide your own algorithm
- Understand the process for using Apache Spark to interact with SageMaker

Algorithms

Algorithms Concepts

Algorithm - Unambiguous specification of how to solve a class of problems

Algorithm vs Heuristic:

Algorithm - Set of steps to follow to solve a specific problem intended to be repeatable with the same outcome

Heuristic - A mental short-cut or "rule of thumb" that provides guidance on doing a task but does not guarantee a consistent outcome

---

	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Training	Training Data and Testing Data	No Training	How to Maximize Reward
Discrete	Classification	Clustering	Simulation-based Optimization
Continuous	Regression	Reduction of Dimentionality	Autonomous Devices

Supervised Learning - A teacher or parent **supervises** the learning process by providing model examples and feedback on quizzes

Unsupervised Learning - You are totally **unsupervised** and must rely on alternative methods to learn other than prior experience.

Reinforcement Learning - A model seeks to maximize reward, often through trial and error. We want to **reinforce** the desired behavior.

How to access ML algorithms in AWS

- Use SageMaker built-in algorithms
- Purchase from AWS Marketplace
- Build your own via Docker image

Regression

Linear Learner Algorithm

Linear models are supervised learning algorithms for regression, binary classification or multiclass classification problems. You give the model labels (x,y) with x being high-dimensional vector and y is a numeric label. The algorithm learns a linear function, or, for classification problems, a linear threshold function, and maps a vector x to an approximation of label y.

To use this algorithm you need a number or list of numbers which yields some other number... the answer you're after. You can use it to predict a specific value or a threshold for grouping purposes.

Adjusts to minimize error

- The algorithm wants the equation to be as good of a fit as possible... meaning the sum of all the distances from the training data point and the line is as low as possible. One method is **Stochastic Gradient Descent**
- Stochastic Gradient Descent - trying to minimize error by finding its way down a slope as quickly as possible. May end up in a local minimum, or go all the way to the global minimum

Things to know about Linear Learner

- Very Flexible
  - Can be used to explore different training objectives and choose the best one. Well suited for discrete or continuous inferences
- Built-in Tuning
  - Has an internal mechanism for tuning hyperparameters separate from the automatic model tuning feature
- Good First Choice
  - If your data and objective meets the requirements, Linear Learner is a good first choice to try for your model

Use Cases

- Predict quantitative value based on given numeric input
  - Example: Based on last five years ROI from marketing spend, what can we expect to be this year's ROI?
- Discrete Binary Classification Problems
  - Example: Based on past customer response, should I mail this particular customer? Yes or No?
- Discrete Multiclass Classification Problems
  - Example: Based on past customer response, how should I reach this customer? Email, Direct Mail, or Phone Call?

Factorization Machines are better for a sparse dataset (one with missing values)

Factorization Machines Algorithm

General purpose supervised machine learning algorithm for both binary classification and regression. Captures interaction between features with high dimensional sparse datasets.

To use this algorithm you need a number or list of numbers which yields some other number... the answer you're after. You can use it to predict a specific value or a threshold for placing into one of two groups. It is a good choice when you have "holes" in your data.

Things to know about Factorization Machines Algorithm

- Considers only pair-wise features
  - Amazon SageMaker's implementation of factorization machines will only analyze relationships of two pairs of features at a time
- CSV is not supported
  - CSV is not a choice for sparse dimensions. File and Pipe mode training are supported using recordIO-protobuf format with Float32 tensors
- Doesn't work for Multiclass Problems
  - Factorization Machines algorithm can be run in either binary classification mode or regression mode
- Really needs LOTS of data
  - To make up for the sparseness, needs lots of data. Recommended dimension of the input feature space is between 10,000 and 10,000,000
- CPUs rock sparse data better
  - AWS recommends CPUs with factorization machines for the most efficient experience
- Don't perform well on dense data
  - Other algorithms are much more performant when you have a full set of data

Use Cases

- High Dimensional Sparse Data Sets
  - Example: Click-stream data on which ads on a webpage tend to be clicked given known information about the person viewing the page
- Recommendations
  - Example: What sort of movies should we recommend to a person who has watched and rated some other movies?

Clustering

What is a Clustering?

- Unsupervised algorithms that aim to group things such that they are with other things more similar than different

K-Means Algorithm

Unsupervised algorithm that attempts to find discrete groupings within data, where members of a group are as similar as possible to one another and as different as possible from members of other groups. The Euclidean distance between these points represents the similarity of the corresponding observations

K-Means will take in a list of things with attributes. You specify which attributes indicate similarity and the algorithm will attempt to group them together such that they are with other similar things. "Similarity" is calculated based on the distance between the identifying algorithms

Things to know about SageMaker K-Means

- Expects tabular data
  - Rows represent the observations that you want to cluster and the columns represent attributes of the observations
- Define the Identifying Attributes
  - You must know enough about the data set to propose attributes that will define similarity. If you have no idea, there are ways around this too!

- SageMaker uses a Modified K-Means
  - AWS uses a modified version of the web-scale K-Means algorithm, which it claims to be more accurate
- CPU Instances Recommended
  - GPU instances can be used but SageMaker's K-Means can only use one GPU
- Training is still a thing
  - You want to be sure your model is still accurate and using the best identifying attributes. Your data just doesn't have labels
- You define number of features and clusters
  - You must define the number of features for the algorithm to analyze and the number of clusters you want

Use Cases

- Pulse Code Modulation - sampling audio, converting it to 1's and 0's
- Handwriting recognition - clustering similar looking images

Classification

K-Nearest Neighbor

Supervised

An index-based, non-parametric method for classification or regression. For classification, the algorithm queries the k points that are closest to the sample point and returns the most frequently used label as the predicted label. For regression, the algorithm queries the k closest points to the sample point and returns the average of the feature values as the predicted value

Predicts the value or classification based on that which you are closest. It can be used to classify or to predict a value (average value of nearest neighbors)

You choose the number of "neighbors"

- You include a value for k, or in other words, the number of closest neighbors to use for classifying

KNN is a *lazy* algorithm

- Does not use training data points to generalize but rather uses them to figure out who's nearby

The training data stays in memory

- KNN doesn't "learn" but rather uses the training dataset to decide on similar samples

Use Cases

- Credit Ratings
  - Example: Group people together for credit risk based on attributes they share with others of known credit usage
- Product Recommendations
  - Example: Based on what someone likes, recommend similar items they might also like

Beware though...KNN is a method of stereotyping and can come with a good degree of bias

Redlining - the practice of literally drawing lines around neighborhoods and classifying those as: A. Best B. Still Desirable C. Definitely Declining D. Hazardous

Private and public entities would then use those redline maps to deny home loans, insurance, and other services for those less desirable areas -- regardless of the qualifications of the applicant

Image Analysis

Amazon Rekognition does this for us

Algorithms:

- Image Classification - Supervised
  - Determine the classification of an image. It uses a convolutional neural network (ResNet) that can be trained from scrath or make use of transfer learning.
  - Think: Hotdog / Not Hotdog
  - Good resource is ImageNet, a huge database of labeled images
- Object Detection - Supervised
  - Detects specific objects in an image and assigns a classification with a confidence score
  - Think: There's a clock and a lamp
- Semantic Segmentation - Supervised
  - Low level analysis of individual pixels and identifies shapes within an image
  - Think: Edge detection
  - Accepts PNG File Input
    - You can submit files for training or inference in uncompressed PNG format
  - Only supports GPU instances for training
    - Due to the heavy computational power required, GPU instances must be used for training
  - Can deploy on either CPU or GPU instances
    - After training is done, model artifacts are output to S3. The model can be deployed as an endpoint with either CPU or GPU instances
  - Good resource is Cityscapes dataset which captures many elements of cities and classifies many elements of those cities, such as roads, pedestrians, motorcycles, etc.

Use Cases

- Image Metadata Extraction
  - Example: Extract scene metadata from an image provided and store it in a machine-readable format
- Computer Vision Systems
  - Example: Recognize orientation of a part on an assembly line and, if required, issue a command to a robotic arm to re-orient the part.

Anomaly Detection

Random Cut Forest

Unsupervised

Detect anomalous data points within a set, like spikes in time series data, breaks in periodicity or unclassifiable points. Useful way to find outliers when it's not feasible to plot graphically. RCF is designed to work with n-dimensional input

Find occurrances in the data that are significantly beyond "normal" (usually more than 3 standard deviations) that could mess up your model training

Gives an Anomaly Score to data points

- Low scores indicate that a data point is considered "normal" while high scores indicate the presence of an anomaly

Scales Well

- RCF scales very well with respect to number of features, data set size and number of instances

Does not benefit from GPU

- AWS recommends using normal compute instance (ml.m4, ml.c5)

Use Cases

- Quality Control
  - Example: Analyze an audio test pattern played by a high-end speaker system for any unusual frequencies
- Fraud Detection
  - Example: If a financial transaction occurs for an unusal amount, unusual time or from an unusual place, flag the transaction for a closer look

IP Insights

Learns usage patters for IPv4 addresses by capturing associations between IP addresses and various entities such as user IDs or account numbers

Can potentially flag odd online behavior that might require closer review

Ingests Entity/IP address Pairs

- Historic data can be used to learn baseline patterns

Returns inferences via a score

- When queried, the model will return a score that indicates how anomalous the entity/IP combination is, based on the baseline

Uses a Neural Network

- Uses a neural network to learn latent vector representations for entities and IP addresses (learns patterns...)

GPUs recommended for training

- Generally, GPUs are recommended but if the dataset is large, distributed CPU instances might be more cost effective

CPUs recommended for inference

- Does not require costly GPUs for normal inference activities

Use Cases

- Tiered Authentication Models
  - Example: If a user tries to log into a website from an anomalous IP address, you might dynamically trigger an additional two-factor authentication routine
- Fraud Detection
  - Example: On a banking website, only permit certain activities if the IP address is usual for a given user login

Text Analysis

Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) algorithm is an unsupervised learning algorithm that attempts to describe a set of observations as a mixture of distinct categories. LDA is most commonly used to discover a user-specified number of topics shared by documents within a text corpus. Here each observation is a document, the features are the presence (or occurence count) of each word, and the categories are the topics

Used to figure out how similar documents are based on the frequency of similar words

Use Cases:

- Article Recommendation
  - Example: Recommend articles on similar topics which might have read or rated in the past
- Musical Influence Modeling

- Example: Explore which musical artists over time were truly innovative and those who were influenced by those innovators

Neural Topic Model

Unsupervised learning algorithm that is used to organize a corpus of documents into topics that contain word groupings based on their statistical distribution. Topic modeling can be used to classify or summarize documents based on the topics detected or to retrieve information or recommend content based on topic similarities

Similar uses and function to LDA in that both NTM and LDA can perform topic modeling. However, NTM uses a different algorithm which might yield different results than LDA.

Sequence to Sequence (seq2seq)

Supervised learning algorithm where the input is a sequence of tokens (for example text, adio) and the output generated is another sequence of tokens

Think a language translation engine that can take in some text and predict what that text might be in another language. We must supply training data and vocabulary

Steps consist of embedding, encoding and decoding

- Using a neural network model (RNN and CNN), the algorithm uses layers for embedding, encoding and decoding into the target

Commonly initialized with pre-trained word libraries

- A standard practice is initializing the embedding layer with a pre-trained word vector like FastText or Glove or to initialize it randomly and learn the parameters during testing

Only GPU instances are supported

- Currently Amazon SageMaker seq2seq is only supported on GPU image types and is only set up to train on a single machine. But it does also offer support for multiple GPUs

Use Cases:

- Language Translations
  - Example: Using a vocabulary, predict the translation of a sentence into another language
- Speech to Text
  - Example: Given an audio "vocabulary", predict the textual representation of spoken words

BlazingText

Supervised & Unsupervised

Highly optimized iomplementations of the Word2vec and text classification algorithms. The Word2vec algorithm is useful for many downstream natural language processing (NLP) tasks, such as sentiment analysis, machine translation, etc.

Really, really optimized way to determine contextual semantic relationships between words in a body of text

BlazingText Modes

Modes	Word2Vec (Unsupervised)	Text Classification (Supervised)
Single CPU Instance	Continuous Bag of Words Skip-gram Batch Skip-gram	Supervised
Single GPU Instance (1 or more GPUs)	Continuous Bag of Words Skip-gram	Supervised with 1 GPU

Modes	Word2Vec (Unsupervised)	Text Classification (Supervised)
Multiple CPU Instance	Batch Skip-gram	None

Expects single pre-processed text file

- Each line in the file should contain a single sentence. If you need to train on multiple text files, concatenate them in one file and upload the file in the respective channel

Highly Scalable

- Improves on traditional Word2Vec algorithm by supporting scale-out for multiple CPU instances. FastText text classifier can leverage GPU acceleration

Around 20x faster than FastText

- Supports pre-trained FastText models but also can perform about 20x faster than FastText

Use Cases:

- Sentiment Analysis
  - Example: Evaluate customer comments in social media posts to evaluate whether they have a postitive or negative sentiment (Amazon Comprehend)
- Document Classification
  - Example: Review a large collection of documents and detect whether the document should be classified as containing sensitive data like personal information or trade secrets (Amazon Macie)

Object2Vec

Supervised

General-purpose neural embedding algorithm that can learn low-dimensional dense embeddings of high-dimensional objects while preserving the semantics of the relationship between the pairs in the original embedding space

A way to map out things in a d-dimensional space to figure out how similar they might be to one another

Word2Vec - feed in words to figure out how closely they are related based on appearances together in different documents

Object2Vec - like word2Vec but doesn't have to be just words, can be products, movies, movie reviews, etc.

Expects pairs of things

- Looking for pairs of items and whether they are "positive" or "negative" from a relationship standpoint. Accepts categorical labels or rating/score-based labels

Feature Engineering

- Embedding can be used for downstream supervised tasks like classification or regression

Training Data Required

- Officially, Object2Vec requires labeled data for training, but there are ways to generate the relationship labels from natural clustering

Use Cases:

- Movie Rating Prediction
  - Example: Predict the rating a person is likely to give a movie based on similarity to other's movie ratings
- Document Classification
  - Example: Determine which genre a book is based on its similarity to known genres (history, thriller, biography, etc.)

Reinforcement Learning

"The carrot and the stick."

Positive Reinforcement - Provide a positive reward thereby motivating the subject to repeat the behavior, presumably for another positive reward

Negative Reinforcement - Provide a displeasurable experience or response thereby motivating the subject to NOT repeat the undesired behavior

Reinforcement Learning (RL) is a machine learning technique that attempts to learn a strategy, called a policy, that optimizes for an agent acting in an environment. Well suited for solving problems where an agent can make autonomous decisions.

Find the path to the greatest reward

Markov Decision Process

Agent - the thing that will do the activity

Agent is placed in an Environment (real or simulated)

Reward - the goal of reinforcement learning is to gain the most reward in the fewest number of steps

State - the information about the Environment and maybe any past steps that might be relevant to any future steps

Action - things the Agent can do

Observation - the information available to the Agent at each State

Episodes - iterations from start to finish that the Agent takes while it's accumulating Reward

Policy - lessons the Agent learns when determining the optimal path to make decisions in the future given a similar objective. What we're after. The decision making part of the Reinforcement Learning model that makes choices to maximize our reward

Use Cases:

- Autonomous Vehicles
  - Example: A self-driving car model can "learn" to stay on the road through iterations of trial and error in a simulation. Once the model is good enough, it can be tested in a real vehicle on a test track.
- Intelligent HVAC Control
  - Example: An RL model can learn patterns and routines of building occupants, impact of sunlight as it transitions across the sky and equipment efficiency to optimize the temperature control for lowest energy consumption

Forecasting

"Past performance is not an indicator of future results."

DeepAR

Supervised

Forecasting algorithm for scalar times series using recurrent neural networks (RNN). DeepAR outperforms standard autoregressive integrated moving average (ARIMA) and exponential smoothing (ETS) by training a single model over multiple time series as opposed to each individual time series

Can predict both point-in-time values and estimated values over a timeframe by using multiple sets of historic data

Cold Start Problem

- Trying to predict the future results of something we don't have historical data for, like sales for a brand new product.
- Can use DeepAR to predict the results for e.g. a new product line by combining the historic sales data of multiple other products

Forecast Type	Example
Point Forecast	"Number of sneakers sold in a week is X"
Probabilistic Forecast	"Number of sneakers sold in a week is between X and Y with Z% probability."

Support for various time series

- Time series can be numbers, counts, or values in an interval (such as temperature readings between 0 and 100.)

More time series is better

- Recommend training a model on as many time series as are available. DeepAR really shines with hundreds of related time series

Must supply at least 300 observations

- DeepAR requires a minimum number of observations across all time series

You must supply some hyperparameters

- Context Length (number of time points the model gets to see before making a predition), Epochs (number of passes over the training data), Prediction Length (the number of time steps that the model is trained to predict, the forecast horizon) and Time Frequency (the granularity of the time series data we are passing in: months, weeks, days, hours, etc.) are required hyperparameters

Automatic Evaluation of the Model

- Uses a backtest after training to evaluate the accuracy of the model automatically

Use Cases:

- Forecasting new product performance
  - Example: Incorporate historical data from other products to create a model that can predict performance of a newly released product.
- Predict labor needs for special events
  - Example: Use labor utilization rates at other distribution centers to predict the required level of staffing for a brand new distribution center

Ensemble Learning

Ensemble Learning - using multiple learning algorithms and models collectively to hopefully improve the model accuracy.

XGBoost

Supervised

Open-source implementation of the gradient boosted trees algorithm that attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models

A virtual "Swiss army knife" for all sorts of regression, classification and ranking problems, with 2 required and 35 optional hyperparameters to tune.

Accepts CSV and libsvm for training and inference

- Uses tabular data with rows representing observations, one column representing the target variable or label and the remaining columns representing features

Only trains on CPU and memory bound

- Currently only trains on CPU instances and is memory-bound as opposed to compute-bound

Recommend lots of memory

- AWS recommends using an instance with enough memory to hold the entire training data for optimal performance

Spark Integration

- Using the SageMaker Spark SDK, you can call XGBoost direct from within the Spark environment

Decision Tree Ensembles Usage Example:

- Determine the right price to sell a house at
- Has many potentially relevant variables
- Can create a decision tree for each independent variable (CART, Classification and Regression Trees)
- Each tree will apply a modifier to the price
- Using multiple trees to create a more realistic estimation model than any one tree could have provided

Use Cases:

- Ranking
  - Example: On an e-commerce website, you can leverage data about search results, clicks, and successful purchases, and then use XGBoost to train a model that can return relevance scores for searched products
- Fraud Detection
  - Example: When XGBoost is given a dataset of past transactions and whether or not they were fraudulent, it can learn a function that maps input transaction data to the probability that transaction was fraudulent

Exam Tips

Concepts:

- Difference between an algorithm and a heuristic
- Be aware of how bias can foul our models
- Understand the difference between a discrete model and a continuous model
- Understand the difference and characteristics of supervised learning, unsupervised learning and reinforcement learning
- Know the options SageMaker provides for algorithms (built-in, buy and bring-our-own)

Regression

- Understand the types of problems best suited for Regression
- Linear Learner algorithm seeks to minimize error via Stochastic Gradient Descent (SGD) with regression problems
- Linear Learner can also be used with classification problems too
- Know that Factorization Machines are best suited for sparse datasets and don't perform well on dense data at all

Clustering

- Know that clustering algorithms are usually unsupervised
- Understand that K-Means can perform clustering similar items based on identifying attributes
- We must define the identifying attributes, number of features and number of clusters

Classification

- Understand the difference between Classification and Clustering
- K-NN can be used for classification or regression problems based on the nearest K data points
- K-NN considered *lazy* algorithm because it does not seek to generalize... rather looks for who's nearest

Image Analysis

- Know that image analysis services are usually classifier models which require training
- Understand the difference between the SageMaker algorithms of Image Classification, Object Detection and Semantic Segmentation
- Be familiar with the higher-level Amazon Rekognition service

Anomaly Detection

- Understand that Random Cut Forest is best suited to detect unusual and out-of-the-ordinary events
- Know that IP Insights is used to detect anomalies between IPv4 addresses and various entities such as user IDs or account numbers

Text Analysis

- Latent Dirichlet Allocation (LDA) most commonly used to figure out similarity of documents but that it also has uses in other clustering problems
- Know that a Neural Topic Model (NTM) and LDA can both perform topic modeling but use different algorithms
- Sequence to Sequence (seq2seq) is often used in language translation and speech to text by using an embedding, encoding and decoding process
- Understand BlazingText is highly optimized and can be used to cluster as well as classify text

Reinforcement Learning

- Know that RL seeks to find the policy that optimizes an agent acting in an environment
- Understand the components of the Markov Decision Process (MDP)
- RL is best suited for situations where the agent can or must make autonomous decisions

Forecasting

- Know why DeepAR is considered to outperform other regression methods of forecasting
- Understand the Cold Start Problem and how DeepAR can help
- Understand the difference between Point Forecasts and Probabilistic forecasts

Ensemble Learning

- Understand Ensemble Learning from a conceptual standpoint
- XGBoost can be used for regression, classification and ranking problems
- Know how XGBoost uses decision trees to create an improvement over linear regression
- XGBoost is "memory-bound" vs "compute-bound"

Evaluation and Optimization

Concepts

Remember: We want **generalization** not **memorization**.

Evaluation Steps:

- Define Evaluation
  - Determine what metric or metrics we should use to decide if the algorithm is "good enough"
- Evaluate
  - Review the metrics during or after the training process. This might be manual or automatic, depending on algorithm
- Tune
  - Adjust hyperparameters, data, the evaluation strategy or even the entire algorithm to bring us closer to the desired results
- Repeat

Types of Validation:

- Offline Validation
  - Validation done using sets of data
  - Example: Validation Sets and K-Fold Validation
- Online Validation
  - Validation under real-world conditions
  - Example: Canary deployments or A/B Testing

Canary Deployment

- Redirect a fraction of traffic to new model, determine whether new version is better than the existing one

Monitoring and Analyzing Training Jobs

Algorithms have two categories of Metrics:

- Training Metrics - Used during the training process to ensure the training is going all right
  - Have a 'test' prefix
- Validation Metrics - Used when testing the model
  - Have a 'validation' prefix
- AWS Documentation describes these as well as Optimization Direction, whether we want this metric to be maximized or minimized

SageMaker sends algorithm metrics and logs are sent to CloudWatch, where they can be viewed and visualized

- SageMaker can also send metrics for custom algorithms to CloudWatch, just have to define the metrics when creating the training job

Evaluation Model Accuracy

Want to avoid Underfitting and Overfitting

Underfitting

Our model just isn't very reflective of the underlying data shape. Maybe we need some more variables to help train our model and achieve a better fit

Methods to prevent Underfitting:

- More Data - sometimes more data will provide enough additional entropy to steer your algorithm away from underfitting
- Train longer - The algorithm neds more iterations with the data to minimize error

Overfitting

Our model is too dependent on that specific data that we used to train. If it sees any new data, accuracy will likely be poor unless the data is identical to the training data. We have trained our model to **memorize** rather than **generalize**.

What we want is a robust model:

- Our model fits the training data but also does reasonably well on new data which it has never seen.
- It can "deal" with noise in the data.
- It can **generalize** effectively for that new data

Training Error	Testing Error	
Low	Low	You want this!

Training Error	Testing Error	
Low	High	Overfitting
High	High	Try another approach
High	Low	Run Away!

Methods to prevent overfitting:

- More Data - sometimes more data will provide enough additional entropy to steer your algorithm away from overfitting
- Early Stopping - Terminate the training process before it has a chance to "overtrain". Many algorithms include this option as a hyperparameter
- Sprinkle in some noise - You training data could be TOO clean and you might need to introduce some noise to generalize the model
- Regulate! - Regularization forces your model to be more general by creating constraints around weights or smoothing the input data
- Ensembles - Combine different models together to either amplify individual weaker models (boosting) or smooth out strong models (bagging)
- Ditch some features - (aka Feature Selection) Too many irrelevant features can influence the model in a negative way by drowning out the signal noise

Regression Accuracy

- If Predicted = 45, Actual = 42, Difference (Residual) = (-3)
- Negative residual means predicted value is higher
- Can graph occurrances of residuals on a histogram to get a residual distribution
    - Want the curve centered around 0, meaning it predicts higher or lower in roughly equal quantities
    - if the curve is centered around a negative number, it's consistently predicting too high
    - centered around a positive number - consistently predicting too low
    - In either the positive or negative center, probably have some systemic flaw in the model

Root Mean Square Error (RMSE)

- RMSE = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(actual_i - predicted_i)^2}$
- A measure of the distance between an actual observation and predicted value
- Lower RMSE is better

Binary Classification Accuracy

	TRUE	FALSE
TRUE	I predicted correctly!	I was wrong. (False Positive) TYPE 1 ERROR
FALSE	I was wrong. (False Negative) TYPE II Error	I predicted correctly!

Area Under the Curve (AUC) is an industry standard for evaluating binary classification models

Recall = $\frac{We\ Guessed\ Right}{We\ Guessed\ Right + We\ Should\ Have\ Flagged\ These\ Too}$

- Example Consequence - Spam Gets Through
- If asked to recall what happened, probably won't have all details, will leave some stuff out

Precision = $\frac{We\ Guessed\ Right}{We\ Guessed\ Right + We\ Flagged\ These\ but\ We\ Were\ Wrong}$

- Example Consequence - Legitimate Emails Get Blocked
- Important that we get 100% of the spam, so have to be very precise

As recall goes up, precision decreases

F1 Score = $\frac{2 * (Precision * Recall)}{(Precision + Recall)}$

- The balance between Precision and Recall
- A larger value indicates better predictive accuracy
- If you want to reduce false positives or false negatives specifically, probably won't optimize for the F1 score

For a multiclass classification problem, can calculate the F1 score for each class and average them to get the Macro Average F1 Score, which represents the overall accuracy of the model.

Improving Model Accuracy

- Collect Data
  - Increase the number of training examples available to the model. More (good) data usually means a more accurate model.
- Feature Processing
  - Provide additional quality variables or refine the existing variables so they are more representative
- Model Parameter Tuning
  - Adjust the hyperparameters used by your training algorithm

BEWARE OF BIAS AND OVER-FITTING

Model Tuning

Model Tuning - Making small adjustments to hyperparameters to improve the performance of the model

Automatic Model Tuning - Also known as hyperparameter tuning, finds the best version of a model by running many jobs that test a range of hyperparameters on your dataset

- Choose a tunable hyperparameter
  - Decide which hyperparameter you want to adjust. Not all hyperparameters can be auto-tuned
- Choose a range of value
  - Specify the range of values to use for tuning the hyperparameter, paying attention to the allowable max/min
- Choose the objective metric
  - Specify the objective metric that the auto-tuning job will seek to optimize

Optimization Metrics:

- Training Metrics - Used during the training process to ensure the training is going all right
  - Have a 'test' prefix
- Validation Metrics - Used when testing the model
  - Have a 'validation' prefix
- AWS recommends different metrics to use based on the algorithm
  - e.g. Use Validation Metrics as the objective metric to avoid overfitting for the Linear Learner
- Recommended metric may differ based on how we're using the algorithm
  - e.g. For BlazingText
    - train:mean_rho for Word2Vec
    - validation:accuracy for Text Classification
- Tunable hyperparameters also differ based on the mode of the algorithm being used

Bayesian Reasoning

- If the value we were looking for was 99%, we could get there in a few ways
  - Matrix - trying ascending values until one works
  - Random - trying random values until one works
  - Both are inefficient
  - Bayesian reasoning takes into account what has happened in the past to build a probability model for what we should choose as the next value

Example:

Hyperparater to Tune - learning_rate

Range - 0.00001 to 1

Optimization Metric - validation:objective_loss (minimize)

learning_rate	objective_loss
0.00001	1.0
0.00010	0.4
0.00100	0.1
...	

Tries to avoid costly iterations by focusing on what hyperparameter value is likely to give more improvement

Exam Tips

Concepts

- Remember we want to Generalize...not Memorize
- Understand the difference between Offline Validation and Online Validation
- Know conceptually about a Canary deployment

Monitoring and Analyzing Training Jobs

- Know the difference between Training metrics and Validation metrics
- Know that CloudWatch integrates well with SageMaker for both logging and metric charting and dashboarding
- Understand how to define custom metrics for custom algorithms

Evaluating Model Accuracy

- Understand underfitting and overfitting as well as potential causes and countermeasures
- Know that regression accuracy is measured by RMSE
- Be able to explain what it means if a histogram of residuals is skewed negatively or positively
- For binary classification, know what AUC metric indicates
- Understand trade-offs and how different scenarios might call for different optimizations
- Understand how to read a confusion matrix and know what an F1 Score and Macro Average F1 Score metric can tell you
- Know some ways to improve model accuracy

Model Tuning

- Recall hyperparameters and how they can be adjusted to control your learning process
- Know the three components you must define for automatic tuning

- Understand that the best metrics to optimize varies by algorithm and sometimes is specific to how you use the algorithm (BlazingText for example)
- Conceptually understand Bayesian Optimization
- Automatic Tuning is not the perfect solution and sometimes may result in weaker models

Implementation and Operations

Concepts

Two Types of Usage

	Offline Usage	Online Usage
What	Make inferences on datasets in batch and return results as a set.	Make inferences on demand as the model is called and return results immediately
Why	Entire dataset is needed for inferences; Pre-process data before using as an input for another model	Need instance response when endpoint is called via an app or service
When	- Predictive models with large historic dataset inputs - Feature engineering for a follow-on model	- Real-time fraud detection - Autonomous machines