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