AWS Certified Machine Learning – Specialty

Release 1.0

Jose Antonio Alvarez Cubero

Dec 31, 2020

Contents

CHAPTER 1

Contents:

1.1 Preface

1.1.1 Authors

- Jose Antonio Alvarez Cubero

Who Should Read This Book?

Expected Audience

The intended audience for this book is those with a general need to understand main AWS services. While AWS certification candidates may experience the largest benefit from this content, the materials included herein may be of use to a much wider audience, especially given modern industry trends towards AWS cloud computing technologies.

There are many elements in this book that explore topics outside the typical contents of AWS certifications. For example the use of AWS python SDK can be viewed as a development-oriented task, however has specific relevance to cloud computing configuration and function, taking a very different approach than traditional CLI-based interface configuration.

Organization of this Book

1.2 ML Building Blocks: Services and Terminology

1.2.1 Introduction to AWS Machine Learning Services

AWS Machine Learning Services

The machine learning stack has 3 key layers:

- Frameworks and Infrastructure.

- Machine Learning platforms.
- API-driven services

The bottom layer provides the framework and infrastructure for machine learning experts who are comfortable building deep learning models, training them, doing inference, and getting the data from their models into production applications. For this group, the vast majority of machine learning and deep learning that's done in the cloud is running on AWS.

The optimized GPU-based instances and the Deep Learning AMI are specifically designed to help expert practitioners get started quickly using their preferred framework. Choice is important, which is why the Deep Learning AMI has all the major frameworks installed, including:

- *Apache MXNet*, which is great for recommendation engines.
- *Caffe* and *Caffe 2*, which are popular for computing vision projects.
- *Tensorflow*.

The second layer of the stack is for customers who want a fully managed platform for building models using their own data. Examples are:

- Apache Spark on Amazon EMR.
- SparkML.

The top layer of this stack is for developers and organizations who want to add intelligence to their applications using an API call rather than developing and training their own models. API based tools are the following:

- *Amazon Rekognition* allows developers to build image analysis directly into their applications, helping them extract rich meta data from visual content. This can be used to perform facial recognition, which allows companies to automatically tag their video content, and it works whether content is archived or freshly ingested.
- *Amazon Polly* allows customers to easily turn text into lifelike speech across many different voices and many different languages.
- *Amazon Lex* which helps developers build intelligent conversational interfaces for existing and new applications. Lex uses the same automated speech recognition and natural language understanding technology that fuels Amazon Alexa. Alexa is often deployed to save cost by automating high frequency interactions with customers and employees, while preserving a high quality user experience.

It's important to remember that machine learning is not an isolated capability but part of an application architecture that builds on the quality of the underlying data. To be good at machine learning, you need a data store that is scalable, available, secure, and flexible. This data lake must handle extremely large data sets, and Amazon S3 is the AWS storage option that hosts the vast majority of data that is stored in the cloud. Once data is in S3, it becomes available for analysis, and AWS provides a broad range of options for data analytics, with services like Amazon Athena, Amazon Redshift, and Redshift Spectrum. Amazon EMR provides access to frameworks like Apache Spark, Presto, Hive, and Pig.

1.2.2 Machine Learning terminology and process

Terminology

Training refers to how the machine learning use historical datasets to build their predictions algorithms. This algorithms make up the **model**, which is the core of the machine learning process. It is what your machine creates after it has been trained and refines over time as it learns. It is what enable the machine to determine output variable from input data, which is then used to generate a **prediction**. A prediction is the machine best estimate for what would be the outcome of a given input or set of inputs would be. It is sometimes called the **inference** of your model.

In a typical traning process, the historical data is used to build the model is splitted into 2 datasets: most of data is used as the training dataset and the remainder is used as the test dataset. The training dataset is inputted into the machine, evaluated by it, and used by the machine to create the first model. The test dataset is then used to test the model for accuracy: It is inputted into the model and the output the model produces from their input is compared with the actual output stored in the test dataset. The close the model predictions are to the real world results in the test dataset the more accurate the model likely is. It is important to understand that training is one of the more important parts of the machine learning process but it is not the whole picture.

Process

Machine learning starts with a business problem and results in a prediction. The goal of machine learning is to address a business problem and this happens by providing accurate predictions. If predictions are not accurate, the machine learning has to get better by providing feedback that improves the way it handles input data and the way to identify features in the data. The key stages of the machine learning process are the following:

ML Problem Framing from the business problem

This is the point in the project in which we will decide what we will actually use and how to use it. This is the place to ask questions like *Do we have all the data needed to answer the business question? What algorithm do we use to answer this business question?* Depending on the amount of data and whether we have a dataset with previously known answers, we can find which type of learning we can do. There are 3 common types of machine learning algorithms: Supervised, unsupervised and reinforcement. In a supervised learning algorithm we will learn from the historical dataset with the answers already known and we will feed them in the machine learning algorithm to predict the future data points. In unsupervised learning algorithms the outcome isn't kwnown beforehand so we let the machine learning algorithm choose how to quantify the data and give us a result. In reinforcement learning the algorithm is rewarded based on the choices it made while learning.

In classification problems, we categorized object into fixed categories. A binary classification problem is when there are only 2 classes in the data. A multiclass classification problem is when there are more than 2 classes.

To frame the machine learning problem, we need to find the output of the problem to evaluate. To structure that output we can use attributes from our dataset: this are called observations. We can also predict future outcomes and this attribute is called the label. To predict the labels from unseen data we will use all other attributes in the dataset, which are called features.

Develop the dataset

If we have that one dataset, we can start integrating the data in one dataset or we can collect all the datasets if there are not in one location, to begin with. We can collect data from multiple sources Amazon S3, Amazon DynamoDB, Amazon Redshift or can be collected simply from the Internet. Wherever the data comes from only it needs to be collected and integrated.

The collected data can be classified into 3 distinct types:

- *Structured* data is data that is organized and stored in databases in the form of rows and columns. This makes query and analysis easy but structured data only makes a chunk of the total available data.
- *Semi-structured* data is data that is typically organized in one the familiar formats but not in tables. This formats include csv, JSON, and more.
- *Unstructured* data is the data that does not have any structure. It is by far the most common type of data. Examples include logs generated by applications servers, text, video and music content among many others.

Data Preparation

At this point, we have our data in one place but is not ready to use to train a model. The data found in the real world is dirty and noisy, meaning it has been improperly collected or formated or it is incomplete, irrelevant or misleading. Because of this, data has to go to some steps before it can be used from training. Putting our data through these steps results in better accuracy and predictions. In preparing our data, we also convert it in an appropriate input format for ingestion into a machine learning algorithm. We can aldo add headers to the columns and convert the columns types.

The presence of missing feature values and outlier can hurt our model performance. The model can suffer because of these data points in addition to the possibility of noisy data or attributes that have missing values. We should clean these ones. We can transform data points using various techniques. Here are some strategies to handle missing values and outliers:

- Introduce a new indicator variable that tells us which represent a missing value.
- Remove the rows that has missing values.
- Imputeation to fill up the missing values. This techniques utilizes the best guess of what the data likely is. It replaces a missing value with a value from the dataset which may be a calculated guess for the data point. For example, if the missing attribute is numerical, you can replace it with the mean or the median. When it is done correctly, imputation immproves the model performance dramatically.

We do not want to model to learn anything based on the order of the data presented, so it is common practice to shuffle the training dataset. This generalizes the model, which improves its quality and its predictive performance. It minimizes the risk of cross validation data under representing the model data and model data not learning from all type of data.

```
train_data = train_data.sample(frac=1)
```

The goal of the machine learning model is to generalize a use case based on the training data that it learns from. From this examples, the model has to predict new examples accurately. To do this, we hold out some of the data from the original dataset: this is splitting the data. The split is generally 80% train and 20% test or 70% train and 30% test. Spliting the data is a validation technique called cross-validation. We can further split the training data into even smaller validation test dataset so that we can evaluate how the model is doing while training and tuning the hyperparameters.

The cross-validation technique can consists of 3 different types:

1. *Validation* where the data is split into train and test datasets. It keeps a lot of data as unseen.
2. *Leave-one-out (LOO CV)*. We only use 1 data point as a test sample and we run the training with the other examples. It‘s worth noting that this technique is computationally expensive.
3. *K-fold*. We randomly split the cases into K-folds and for each fold we train the model and record the error.

1.2.3 Amazon SageMaker

1.2.4 Amazon Comprehend

1.3 Data Science

1.3.1 What is data science?

Data Science is a set of processes and systems to extract knowledge or insights from data, either structured or unstructured (Wikipedia). For the purpose of this document consists of managing, analyzing and visualizing data in support of machine learning workflow.

1.3.2 What is machine learning?

Machine learning: Artificial intelligence machines that improve their predictions by learning from large amounts of input data.

Main idea: Learning = estimating underlying function f by mapping data attributes to some target value.

Training set: A set of labeled examples $(x, f(x))$ where x is the input variables and the label $f(x)$ is the observed target truth.

Goal: Given a training set, find approximation $\hat{f}$ of f that best generalizes, or predicts, labels for new examples. Best is measured by some quality measure, for instance: error rate, sum squared error.

Image:ML.png

1.3.3 Why machine learning

Difficulty in writing some programs

- Too comples (facial recognition)
- Too much data (stock market predicions)
- Information only available dynamically (recommendation system)

Use of data for improvement

- Humans are used to improving based on experience (data)

A lot of data available

- Product recommendations
- Fraud detection
- Facial recognition
- Language understanding

1.3.4 Types of machine learning

Supervised learning

A "teacher" provides training examples, each with correct label. Regression y classification.

Unsupervised learning

Correct label not available for training examples, must find patterns in data (e.g. using clustering). Example: grouping customers according to what books and movies they like.

Reinforcement learning

Not told what action is correct, but given some reward or penalty after each action in a sequence. Example: learning how to play soccer

semi-supervised learning,

1.3.5 Data matters

image:data.png

- Unleash the business value in data collected
- Prepare you to do data science projects and to implement production systems
- Predict future events based on past data leading to proactive change than reactive

1.3.6 The Data Science and ML workflow

Image:workflow.png

Concepts:

Dataset

Trainig set versus test set

Feature = attribute = independent variable = predictor

Label = target = outcome = class = dependent variable = response

Dimensionality = numer of features

Model selection

1.3.7 Key Issues in ML

Data quality

Consistency of the data, Accuracy of the data, noisy data, missing data, outliers in the data, Bias, Variance

Model quality

Image:modelquality.png

Overfitting

Failure to generalize: Model performs well on trainig set but poorly on test set.

Typically indicates that model is too flexible for amount of training data.

Flexibility allows it to “memorize” the data, including noise.

Corresponds to high variance - small changes in the training dat lead to big changes in the results

Underfitting

Failure to capture important patterns in the training data set.

Typically indicates that model is too simple or there are too few explanatory variables.

Not flexible enough to model real patterns

Corresponds to high bias - the results show systematic lack of fit in certain regions

Computation speed and scalability

Use distributed computing systems like Amazon SageMaker or Amazon EC2 instances for training in order to: increase speed, solve prediction time complexity, solve space complexity.

1.4 Machine learning algorithms

1.4.1 Supervised methods

Linear regression

Linear methods are parametric methods where function learned has form $f(x) = \phi\left(w^T x\right)$ where $\phi()$ is some activation function.

Generally, optimized by learning weights by applying (stochastic) gredient descent to minimize loss function, e.g. $\sum|\hat{y}_i - y_i|^2$

Simple; a good place to start for a new problem, at least a baseline

Methods: Linear regression for numeric target outcome. Logistic regression for categorical target outcome.

Linear regression (univariate)

Image:univariate.png

Model relation between a single feature (explanatory variable x) and a real-valued response (target variable y)

Given data (x, y) and a line defined by w_0 (intercept) and w_1 (slope), the vertical offset for each data point from the line is the error between the true label y and the prediction based on x

The best line minimizes the sum of squared errors (SSE)

We usually assume the error is Gaussian distributed with mean zero and fixed variance

Linear regression (multivariate)

Multiple linear regression includes N explanatory variables with $N \geq 2$:

$$y = w_0 x_0 + w_1 x_1 + \cdots + w_N x_N = \sum_{i=0}^{N} w_i x_i$$

Sensitive to correlation between features, resulting in high variance of coefficients.

scikit-learn implementation:

```
sklearn.linear_model.LinearRegression
```

Logistic regression

Predict whether a credit card transaction is fraud

Image:fraud.png

Estimates the probability of the input belonging to one of the two classes: positive and negative.

Vulnerable to outliers in training data.

Relation to linear model:

$$\sigma(z) = \frac{1}{1 + \mathrm{e}^{-z}}$$

z is a trained multivariate linear function

ϕ is a fixed univariate function (not trained)

Objective function to maximize = probability of the true training labels.

Sigmoid curve. Image: sigmoid.png

Model relation between features (explanatory variables x) and the binary responses ($y = 1$ or $y = 0$)

For all features, define the linear combination:

$$z = w^T x = w_0 + w_1 x_1 + \cdots + w_N x_N$$

Define the probability of $y = 1$ given x as p and find the logit of p as:

$$logit(p) = \log \frac{p}{1 - p}$$

Logistic regression finds the best weight vector by fitting the training data

$$logit(p(y = 1|x)) = z$$

Then, for a new observation, you can use the logistic function $\phi(z)$ to calculate the probability to have label 1. If it is larger than a threshold (for example 0.5), you will predict the label for the new observation to the positive.

```
sklearn.linear_model.LogisticRegression
```

Linear separable versus non-linearly separable

Image:separable.png

1.5 Analytics Services

AWS serverless data analytics pipeline reference architecture

1.5.1 Amazon Athena

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL expressions. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries you run. Athena is easy to use. Simply point to your data in Amazon S3, define the schema, and start querying using standard SQL expressions. Most results are delivered within seconds. With Athena, there?s no need for complex ETL jobs to prepare your data for analysis. This makes it easy for anyone with SQL skills to quickly analyze large-scale datasets.

Athena is serverless, so there is no infrastructure to set up or manage, and you pay only for the queries you run. Athena scales automatically?executing queries in parallel?so results are fast, even with large datasets and complex queries.

Athena helps you analyze unstructured, semi-structured, and structured data stored in Amazon S3. Examples include CSV, JSON, or columnar data formats such as Apache Parquet and Apache ORC. You can use Athena to run ad-hoc queries using ANSI SQL, without the need to aggregate or load the data into Athena.

1.5.2 AWS Data Pipeline

1.5.3 Amazon Elasticsearch

1.5.4 Amazon EMR

Amazon EMR provides a managed Hadoop framework that makes it easy, fast, and cost-effective to process vast amounts of data across dynamically scalable Amazon EC2 instances. It securely and reliably handles a broad set of big data use cases, including log analysis, web indexing, data transformations (ETL), machine learning, financial analysis, scientific simulation, and bioinformatics. You can also run other popular distributed frameworks such as Apache Spark, HBase, Presto, and Flink in Amazon EMR, and interact with data in other AWS data stores such as Amazon S3 and Amazon DynamoDB.

1.5.5 AWS Glue

What is AWS Glue?

1.5.6 Amazon Kinesis streaming data platform

Amazon Kinesis streaming data platform consists of Kinesis Data Firehose, Kinesis Data Streams, Kinesis Video Streams, and Amazon Kinesis Data Analytics.

Amazon Kinesis Data Firehose

Amazon Kinesis Data Firehose is the easiest way to load streaming data into data stores and analytics tools. It can capture, transform, and load streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch Service (Amazon ES), and Splunk, enabling near real-time analytics with existing business intelligence tools and dashboards you?re already using today.

It is a fully managed service that automatically scales to match the throughput of your data and requires no ongoing administration. It can also batch, compress, and encrypt the data before loading it, minimizing the amount of storage used at the destination and increasing security.

For Amazon ES destinations, streaming data is delivered to your Amazon ES cluster, and it can optionally be backed up to your S3 bucket concurrently.

Kinesis Data Streams

The vocabulary for data streaming is putting records into a data stream and reading the stream back. You can have multiple shards that provide throughtput. You can create as many shards as you need with a data stream, they are not created dynamically.

Let's image a producer want to send a record, then in order to Kinesis Data Streams what shard to use, you tag the record with the partition key. This partition key is very similar to the message group identifier in a SQS FIFO queue. The process is the following:

1. The producer calls put record.
2. The Kinesis Data Streams service needs to decide to which shard this record is going to be appended. This decision is done through hashing. Each shard owns a subset of hashes. Kinesis calculates the hash for the message partition value and finds out to which shard it belongs.

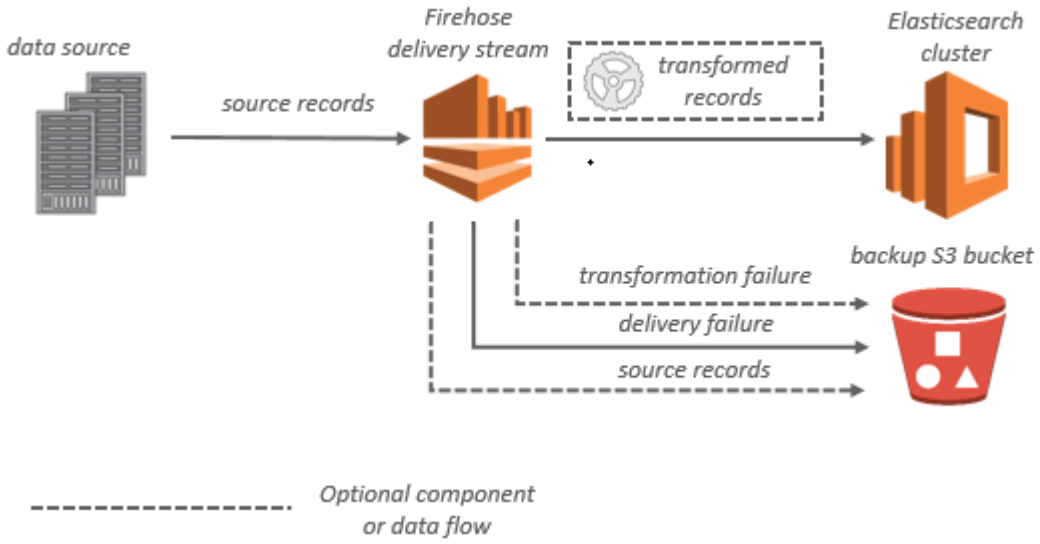

Fig. 1: Amazon Kinesis Data Firehose with Amazon ES destination

3. Amazon Kinesis Data Streams stores the record and synchronously replicates data across three availability zones, providing high availability and data durability.
4. It is return an OK back to the producer.

Image an scenario where:

1. A producer sends a record to the appropriate shard in terms of its shard.
2. The stream stores the record durably to its correspondent shard.
3. There is a networking problem when the producer was calling send message.
4. The producer gets a timeout. It doesn't know if Kinesis got the record or it didn't.
5. The producer retry the send.
6. Kinesis calculate again the record hash and sends it to the same shard as the previous version. There is going to be a duplicate.

When you need more throughput and you have already data in your data stream, you can do a shard split. You select one of the shards and split it into two: you divide the hash space of one the shards into two hash subspaces. You can scale the throughput as much as you want by adding as many shards as you want. Resharding means that it does not influence existing data, the records get appended to the new shards if applicable. You do not have to decide which shards are needed to be splitted, it is done automatically by Kinesis.

A consumer is responsible for deciding from which shard is going to consume and which record to consume. The first action that the consumer should do is to ask Kinesis what shards are available. Then it will a pick a shard and start reading from an iterator. As a consequence, with Kinesis Data Streams there is consumer affinity: a consumer read all the records from a shard. It is easy to perform analysis of consecutive entries.

When you consume from a data stream, you are not deleting any record. This means that you can start multiple applications consuming from the same stream, like a fanout pattern, doing some analysis on the data stream.

If the number of shards increases, the complexity of the code of the consumers increases as well. The consumer is responsible for keeping track who is reading, from which shard, check the progress of reading, ... Because of that, instead of using directly Kinesis API for consuming, it is recommend to use existing client-side libraries called Kinesis

Client Library (KCL). This library does the complex management of shards: electing who is reading from each shard, of tracking progress of your reads, and allows to focus on the code needed to process the data in the records.

A new shard iterator is returned by every GetRecords request (as `NextShardIterator`), which you then use in the next GetRecords request (as `ShardIterator`). Typically, this shard iterator does not expire before you use it. However, you may find that shard iterators expire because you have not called GetRecords for more than 5 minutes, or because you've performed a restart of your consumer application.

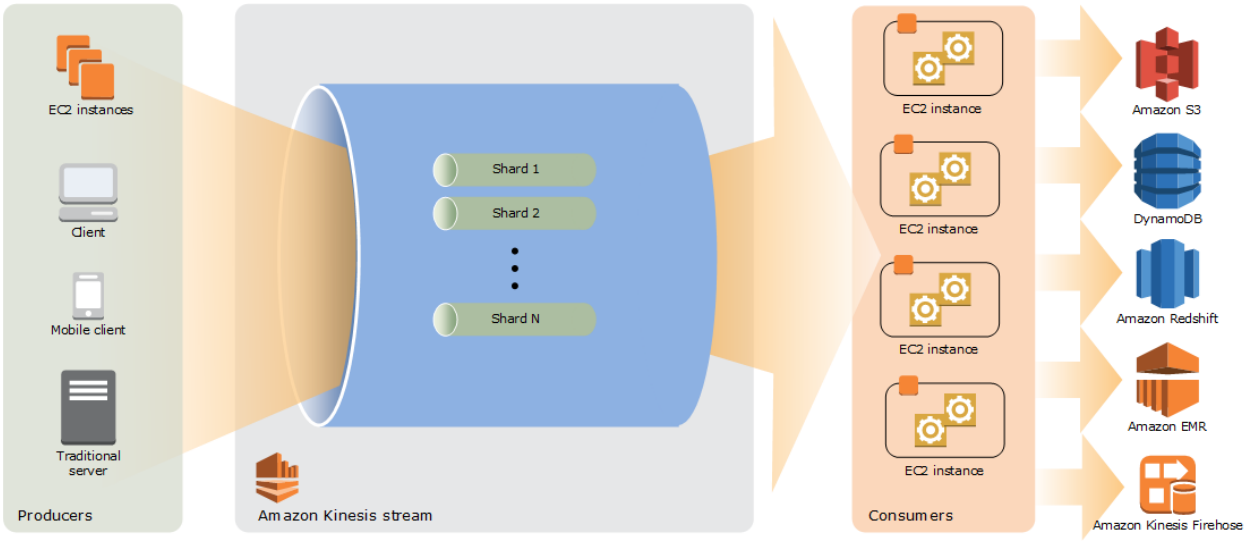

Fig. 2: Kinesis Data Streams architecture

Kinesis Data Streams supports changes to the data record retention period of your stream. A Kinesis data stream is an ordered sequence of data records meant to be written to and read from in real-time. Data records are therefore stored in shards in your stream temporarily.

The time period from when a record is added to when it is no longer accessible is called the retention period. A Kinesis data stream stores records from 24 hours by default to a maximum of 168 hours (7 days).

Resharding

Amazon Kinesis Data Streams supports resharding, which lets you adjust the number of shards in your stream to adapt to changes in the rate of data flow through the stream. Resharding is considered an advanced operation.

There are two types of resharding operations: shard split and shard merge. In a shard split, you divide a single shard into two shards. In a shard merge, you combine two shards into a single shard. Resharding is always pairwise in the sense that you cannot split into more than two shards in a single operation, and you cannot merge more than two shards in a single operation. The shard or pair of shards that the resharding operation acts on are referred to as parent shards. The shard or pair of shards that result from the resharding operation are referred to as child shards.

Splitting increases the number of shards in your stream and therefore increases the data capacity of the stream. Because you are charged on a per-shard basis, splitting increases the cost of your stream. Similarly, merging reduces the number of shards in your stream and therefore decreases the data capacity?and cost?of the stream.

If your data rate increases, you can also increase the number of shards allocated to your stream to maintain the application performance. You can reshard your stream using the `UpdateShardCount` API. The throughput of an Amazon Kinesis data stream is designed to scale without limits via increasing the number of shards within a data stream.

Amazon Kinesis Data Analytics

Amazon Kinesis Data Analytics is the easiest way to analyze streaming data, gain actionable insights, and respond to your business and customer needs in real time. Amazon Kinesis Data Analytics reduces the complexity of building, managing, and integrating streaming applications with other AWS services. SQL users can easily query streaming data or build entire streaming applications using templates and an interactive SQL editor. Java developers can quickly build sophisticated streaming applications using open source Java libraries and AWS integrations to transform and analyze data in real-time.

Amazon Kinesis Data Analytics takes care of everything required to run your real-time applications continuously and scales automatically to match the volume and throughput of your incoming data. With Amazon Kinesis Data Analytics, you only pay for the resources your streaming applications consume. There is no minimum fee or setup cost.

1.5.7 Amazon QuickSight

1.6 Data Engineering

1.6.1 Create data repositories for machine learning

1.6.2 Identify and implement a data-ingestion solution

1.6.3 Identify and implement a data-transformation solution

1.7 Exploratory Data Analysis

1.7.1 Sanitize and prepare data for modeling

- Graphtext

1.7.2 Perform feature engineering

1.7.3 Analyze and visualize data for machine learning

- Google data studio

1.8 Modeling

Amazon SageMaker Automatic Model Tuning: Using Machine Learning for Machine Learning

Amazon SageMaker Workshops

1.8.1 Frame business problems as machine learning problems

1.8.2 Select the appropriate model(s) for a given machine learning problem

1.8.3 Train machine learning models

1.8.4 Perform hyperparameter optimization

1.8.5 Evaluate machine learning models

1.9 Machine Learning Implementation and Operations

1.9.1 Build machine learning solutions for performance, availability, scalability, resiliency, and fault tolerance

1.9.2 Recommend and implement the appropriate machine learning services and features for a given problem

1.9.3 Apply basic AWS security practices to machine learning solutions

1.9.4 Deploy and operationalize machine learning solutions

How to Deploy Deep Learning Models with AWS Lambda and Tensorflow

1.10 Appendix

1.10.1 Data

This Machine Learning Project on Imbalanced Data Can Add Value to Your Resume

24 Ultimate Data Science (Machine Learning) Projects To Boost Your Knowledge and Skills (& can be accessed freely)

1.10.2 EDA

Dataset exploration: Boston house pricing

1.10.3 Libraries

CatBoost

1.10.4 Modeling

Forecasting: Principles and Practice

Interpreting machine learning models

Time Series Analysis in Python: An Introduction

Python Code for Identifying Seasonal Customers

1.10.5 Visualization

Tufte

Brushable Scatterplot Matrix

1.10.6 Method

The toolkit for the modern data ninja

1.10.7 Repositories

Most Active Data Scientists, Free Books, Notebooks & Tutorials on Github

A gallery of interesting Jupyter Notebooks

CHAPTER 2

Indices and tables

- genindex
- modindex
- search