AWS Data Engineering

# Sage maker

# Used for creating Jupiter notebook in AWS

# Data Gathering

* Public Repositories
* Scrapping the webpages

# Data Cleaning - Handling missing values

* Do nothing – (ignore observations)
* Remove the entire record
* Mode – (For Categorical Data)
* Median /average value replacement – (For Numerical Data)
* Most frequent value
* Model – based imputation – (K-Nearest Neighbors (feature similarity)  
  / Regression (using correlation matrix) / Deep Learning)
* Interpolation / Extrapolation – (estimate values from other observations within the range of a discreate set of known data points)
* Forward filling / Backward filling – (filling the missing value from the preceding or succeeding values)
* Hot deck imputation – (randomly choosing the missing value from the set of related and similar variable)

# Feature Extraction

* **Dimensionality** – refers to the number of features in your dataset
* **Reducing** **dimensionality** – refers to reducing the number of features keeping only required ones
* **Feature selection**
* **Feature** **Extraction** –PCA- take features and give something less (reducing the number of features while still retaining as much information as possible
* **PCA**: **principle** **component** **analysis**: unsupervised algorithm that creates new features by linearly combining original features

# Encoding categorical data

Binarize Encoding: for features of a binary nature -- true/false, on/off, male/female etc.

Label Encoding (non binary) : may imply ordinality, can use ordinal encoder.

One – hot -encoding: change nominal categorical values into numerical values

|  |  |  |
| --- | --- | --- |
|  | Is\_sunny | Is\_cloudy |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 2 | 0 | 1 |
| 3 | 1 | 0 |

Example

|  |  |
| --- | --- |
|  | Weather |
| 0 | Sunny |
| 1 | Cloudy |
| 2 | Cloudy |
| 3 | sunny |

# Numeric feature engineering

Transforming numeric values so machine learning algorithms can better analyze them.

Change numeric values so all values are on the same scale.

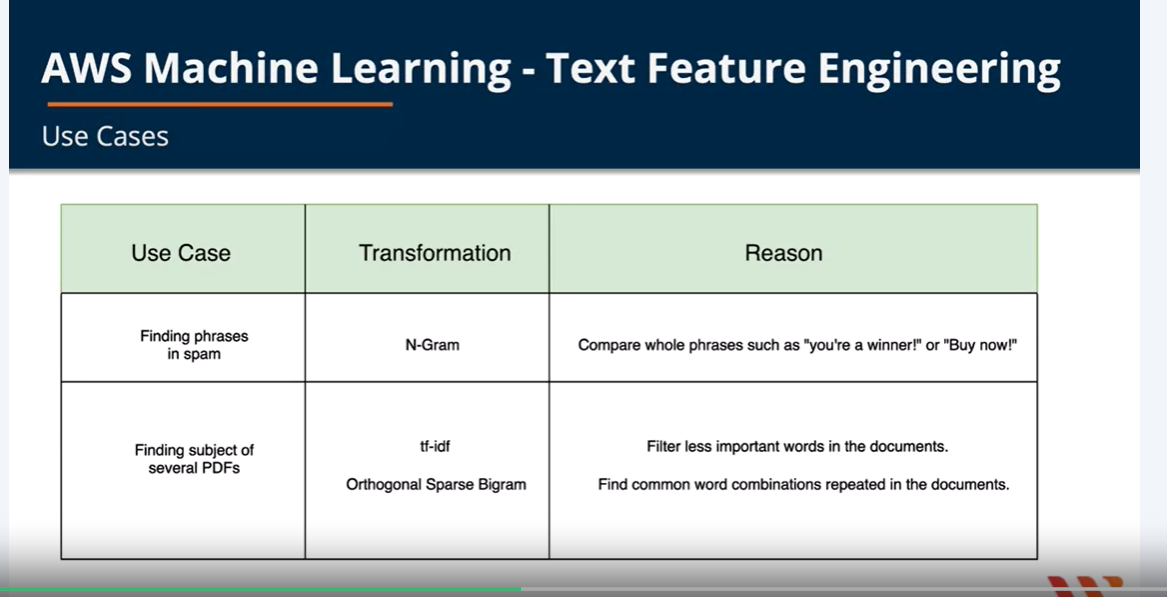
* **Normalization**: rescale the values into a range [0,1] .
* **Standardization**: Rescale data to have a mean of 0 & standard deviation of 1.
* **Binning**: re-organizing continuous data into bins – use qcut in python for it

# Text Feature Engineering

Text Feature Editing : transform text within data so machine learning algorithms can better analyze it.

Splitting text into smaller pieces

* Bag of words
  + tokenizes raw text and creates a statistical representation of the text.
  + Breaks up text by white spaces into single words
* N-Gram
  + Extension of bag of words which produces groups of words of size
  + Breaks up text by whitespace into groups pf words
* Orthogonal Sparse Bigram
  + Creates groups pf words of size n, returns every pair of words that includes the first word
  + Creates groups of words that always include the first word
* Term Frequency-Inverse Document Frequency (tf-idf).
  + Shows how important a word or words are o a given set of text by providing appropriate weights to terms that are common and less common.



# Amazon Mechanical Turk

Amazon Mechanical Turk service

is one of the on demand and

scalable human workforce service for completing the task

where the human capabilities

are generally outperforming the computers.

# AWS Migration Services

* Migrate data from source to machine learning repository
* Several aws services to help move data
  + **Amazon data pipeline:**
    - copy data using pipeline activities
    - schedule regular data movement
  + **Aws database migration services (DMS):**
    - Move data between databases
    - MySQL to MySQL
    - Aurora to DynamoDB
  + **AWS GLUE**
    - ETL SERVICE
  + **Amazon sage maker**
    - Use jupyter notebook
  + **Amazon Athena**
    - Run sql queries on S3 data

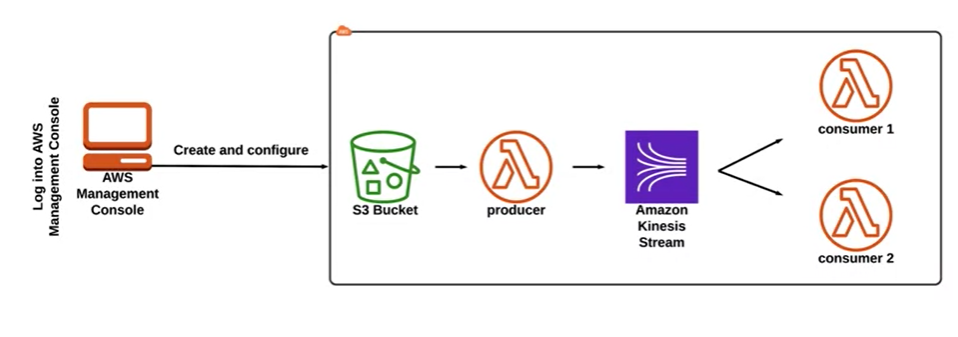
Exploratory data analysis in AWS

# Kinesis Data streams

* Gets data from data producers such as IoT, Social media.
* Uses shards to stream data to consumers such as EC2, lambda, Kinesis Data analytics.
* Consumers then send data to a data repository such as S3, DynamoDB, Redshift, or Business Intelligence Tools.

# Building Realtime Data Streaming System-Kinesis Data Stream

**Create S3 Bucket 🡪 Create Kinesis Data Stream 🡪 Create Producer Lambda Functions**



# Kinesis Data Firehose

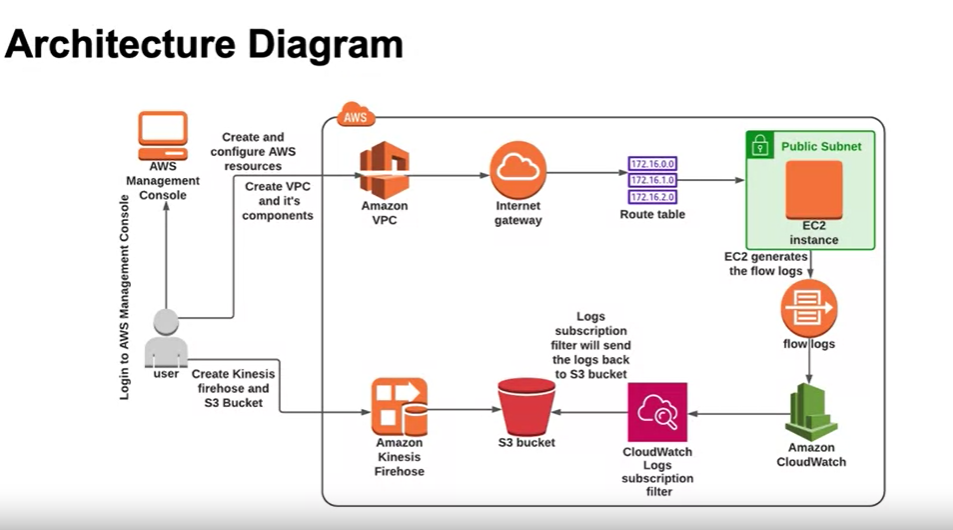
* Receives data from producers such as IoT, social media
* Uses lambda functioning instead of shards to transmit producer data
* Lambda function puts data to data stores such as S3, Redshift, ElasticSearch, or splunk
* Can transmit directly from producers through Firehose to the data store (don’t have to use lambda intermediary)
* S3 events to store to DynamoDB

# Kinesis Data Firehose - Create S3 - VPC - VPC Flow Logs

Create S3 Bucket 🡪 Create Kinesis Firehose delivery Stream 🡪 Create Cloudwatch log group 🡪

Create VPC 🡪 Internet gateway 🡪 Subnet 🡪 EC2 Instance 🡪

🡪 Generate EC2 Logs 🡪 Create Subscription filter



# Kinesis Video Streams

* Build video processing applications such as machine learning models
* Producers such as web cams, security cameras, audio feeds, images
* Data consumers – Kinesis video stream applications
* Stores to S3

# Kinesis Data Analytics

* Use SQL to process streaming data
* Sources: Kinesis Data Streams and Kinesis Data Firehose
* SQL queries put to S3, Redshift, or Visualization and Business Intelligence tools

# EMR – Amazon Elastic MapReduce

* Managed platform designed for cluster-centric workloads
* Amazon EMR is widely used for tasks like log analysis, data transformations, and large-scale machine learning.
* It allows the execution of prominent big data frameworks such as Apache Spark or Apache Hadoop on the AWS cloud, facilitating the processing and analysis of extensive data sets

Layers of Amazon EMR

* Storage -> (HDFS, EMRFS, Local file system)
* Cluster Resource Management -> (Responsible for Cluster Resource Management and scheduling tasks)
* Data Processing Frameworks -> (consists of Hadoop map reduce , Apache Spark)
* Applications and programs -> (helps processing and management of big data sets like HIVE, PGI, Streaming Libraries and ML Algorithms)

Create Amazon EC2 Key Pair 🡪 Create S3 Bucket 🡪 Upload application and input data to S3

🡪 launch EMR Cluster using spark 🡪 submit work to amazon EMR 🡪 validate the results

🡪 SSH to cluster

# Amazon Managed Service for Apache Flink

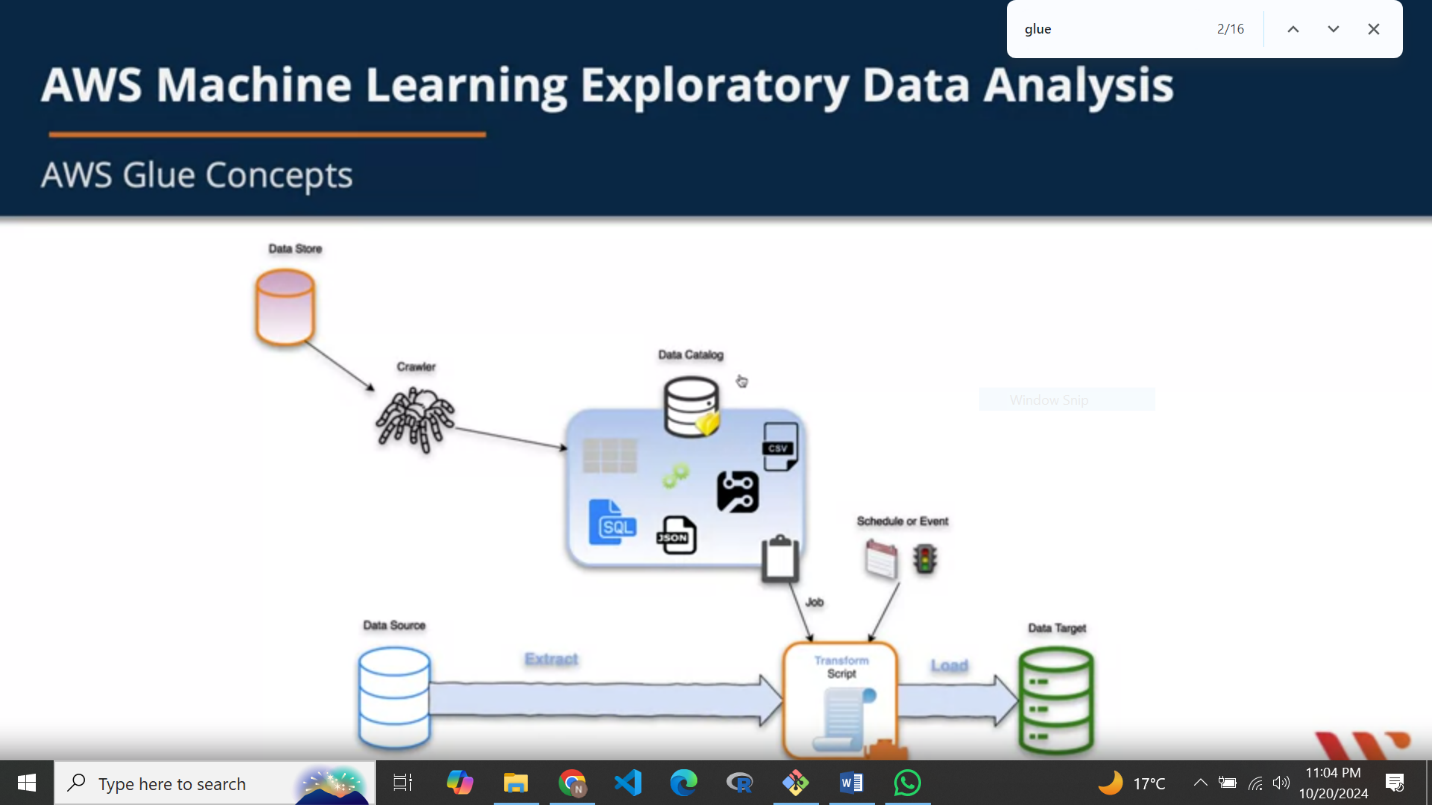
With amazon managed service for apache flink you can utilize java, Scala, or SQL to process and analyze streaming data

Amazon Kinesis Data Streams (Data is sent to amazon kinesis data stream)

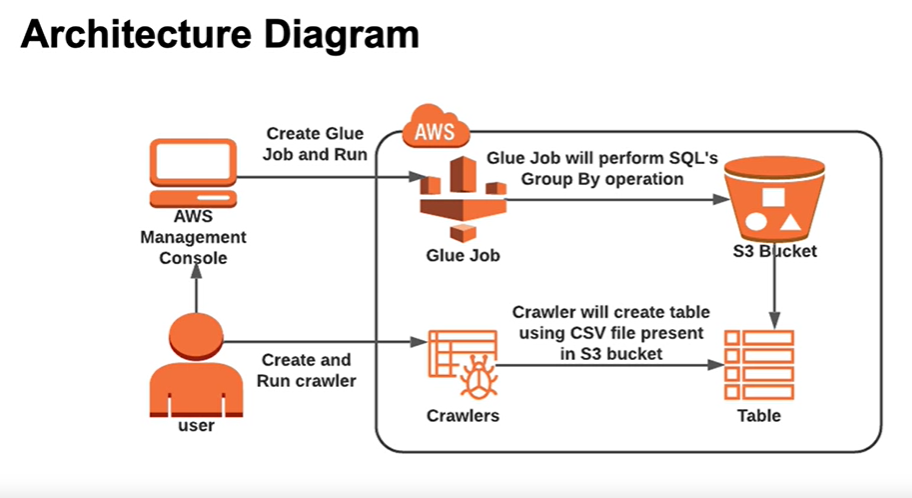
Amazon Managed Service for Apache Flink (Reads the data from kinesis Data Stream and Processes it).

Amazon S3 Destination (Processed data is sent to a destination S3 Bucket)

# AWS Glue Concept



* A fully managed ETL service for categorizing, cleaning, enriching, and moving your data



# Amazon Redshift

* A fully managed data warehouse
* Handles petabytes of data
* fully managed, cloud-based data warehouse service designed for large-scale data analytics. It allows you to run complex SQL queries on structured and semi-structured data across petabytes of data

# Amazon Data Exchange

* **AWS Data Exchange** is a service that makes it easy for users to find, subscribe to, and use third-party data in the AWS cloud.
* It provides a marketplace where data providers can publish and distribute their datasets, while consumers can access and integrate these datasets into their workflows or analytics.

# Amazon Quick sight

Data visualization tool provided by amazon

Modeling in AWS

1. Create a model in sagemaker
2. Create an endpoint configuration
3. Create HTTPS endpoint

# Machine learning models

1. **Supervised Learning 🡪 target variable to be predicted from independent variable** 
   1. Classification
   2. Regression
   3. Examples

|  |  |
| --- | --- |
| Linear Regression | Logistic Regression |
| Gradient Boosted Trees | Nearest Neighbor |
| SVM – Support Vector Machine | Neural Networks |
| Decision trees | Naïve Bayes |

1. **Unsupervised Learning 🡪 Not target or outcome to predict**
   1. Clustering
   2. Dimensionality Reduction

|  |  |
| --- | --- |
| K-means Clustering | PCA – Principle component analysis |
| t-Distributed Stochastic neighbor embedding |  |

1. **Reinforcement Machine Learning 🡪 trained to make specific decisions, trains itself continuously in an environment using trial and error**

|  |  |  |
| --- | --- | --- |
| Q Learning | Temporal difference (TD) | Monte-Carlo Tree Search |
| Asynchronous Actor-Citric Agents (A3c) |  |  |

# Hyperparameters

A hyperparameter is a parameter whose value is set before the learning process

1. **Model hyperparameter**: influence the performance of the model
2. **Algorithm hyperparameter**: affect the speed and quality of the learning process

Two approaches for choosing hyperparameters

1. **Random Search:** chooses a random combination of values for each training job
2. **Bayesian Search**: performs hyperparameter tuning as a regression problem

Machine Learning Algorithms

Sage maker built in algorithms

1. Regression algorithms 🡪 linear regressor
2. Clustering algorithms 🡪 k-means clustering
3. Classification algorithms 🡪 Linear Learner , Text Blazing (Word2vec,text classification), NLP tasks, XGBOOST(regression and classification[use weights of the variables]), K-nearest Neighbors, Factorization matrix-(recommendations)
4. Image analysis algorithms 🡪Resnet 🡪 (eg: classify image as offensive or not on platforms)
5. Text analysis algorithms 🡪 Text Blazing (Word2vec,text classification), NLP tasks,

SequenceTOsequence

1. Anomaly detection algorithms
2. Reinforcement algorithms
3. Forecasting algorithms

## Morkv Decision Processes

A **Markov Decision Process (MDP)** is a mathematical framework used in reinforcement learning and decision-making, where an agent interacts with an environment to make a series of decisions in a stochastic (probabilistic) setting. MDPs are used to model situations where outcomes are partly random and partly under the control of the decision-ma ker.

AWS tools for deploying and maintaining models

1. **SageMaker Studio**: An all-in-one ML environment for building, training, and deploying models with an easy-to-use interface.
2. **SageMaker Autopilot**: AutoML tool that builds and ranks models automatically, ideal for rapid experimentation.
3. **SageMaker Experiments**: Tracks and organizes ML experiments, helping manage and compare different model versions.
4. **SageMaker Debugger**: Detects issues during training (like overfitting) and adds fault tolerance by saving progress and diagnosing errors.
5. **SageMaker Model Monitor**: Tracks data drift and model quality in real-time, alerting you when retraining might be needed to keep models reliable.

Security in AWS

To secure AWS SageMaker instances, here are some essential practices:

1. **IAM Policies**: Use Identity and Access Management (IAM) roles to restrict access, ensuring that only authorized users and applications can access SageMaker resources.
2. **VPC Configuration**: Deploy SageMaker instances within an Amazon Virtual Private Cloud (VPC) to control network traffic and limit access to only trusted networks.
3. **Encryption**: Enable encryption for data at rest (using Amazon S3 and EBS encryption) and in transit (using HTTPS/TLS).
4. **Network Access Control**: Use Security Groups and Network ACLs to define inbound and outbound rules for controlling access to SageMaker instances.
5. **Logging and Monitoring**: Use Amazon CloudWatch to monitor instance activity and enable AWS CloudTrail for logging API requests, which helps detect unauthorized actions.
6. **SageMaker Notebook Lifecycle Configurations**: Automate security checks and environment configurations (e.g., install patches) by configuring notebooks to run secure code at startup.
7. **SageMaker Model Monitor**: Continuously monitor deployed models to detect and alert on unexpected behaviors or data drifts that could indicate security issues.

Following these practices helps maintain strong security for SageMaker instances and data.

AWS IoT Greengrass

is a service that brings cloud capabilities to local devices, allowing them to process data, run applications, and communicate with each other even when not connected to the internet. It’s designed for edge computing, enabling devices to perform tasks locally and only sync with the cloud when needed.

Key features:

* **Edge Processing**: Runs ML models, collects data, and responds to local events in real-time without internet connectivity.
* **Seamless Integration**: Connects with other AWS services, making it easy to manage and update applications across IoT devices.
* **Security**: Uses AWS security tools for device authentication, encryption, and data protection, keeping devices and data secure.

IoT Greengrass is ideal for industrial automation, remote monitoring, and real-time data processing in areas with intermittent connectivity.

Amazon Connect

is a cloud-based contact center service that helps businesses set up customer support operations easily. It allows companies to manage inbound and outbound calls, chats, and tasks with features like IVR (Interactive Voice Response), AI-driven chatbots, and real-time analytics.

Key points:

* **Easy setup**: Quickly deploy a scalable contact center.
* **Omnichannel support**: Handles phone, chat, and messaging on one platform.
* **AI integration**: Uses AWS AI tools for chatbots and speech analytics.
* **Pay-as-you-go**: Only pay for usage, making it cost-effective.

Amazon Connect is ideal for businesses looking to create flexible, scalable, and AI-enhanced customer service solutions.

4o