Up-to-date Questions and Answers from authentic resources to improve knowledge and pass the exam at very first attempt. ----- Guaranteed.

KILL EXAMS

MLS-C01 MCQs
MLS-C01 TestPrep
MLS-C01 Study Guide
MLS-C01 Practice Test
MLS-C01 Exam Questions

killexams.com

Amazon

MLS-C01

AWS Certified Machine Learning Specialty (MLS-C01)

ORDER FULL VERSION

https://killexams.com/pass4sure/exam-detail/MLS-C01

Question: 894
You build an RNN with two stacked LSTM layers (64 units each) in SageMaker to forecast hourly energy usage from a 24-hour sequence. You use tanh activation, a batch size of 32, and a learning rate of 0.01. After 50 epochs, the model predicts flat values across all hours. What's the most likely cause, and how should you fix it?

A. Vanishing gradients; switch to ReLU activation
B. Learning rate too high; reduce it to 0.001
C. Insufficient capacity; add a third LSTM layer
D. Data not normalized; scale inputs to [0, 1]

Answer: B

Explanation: Flat predictions in RNNs often result from a learning rate too high (0.01), causing unstable updates that prevent the model from learning temporal patterns. Reducing it to 0.001 stabilizes training, allowing the LSTM to capture dependencies. Vanishing gradients are mitigated by LSTMs, and normalization or capacity isn't indicated as the primary issue.

Question: 895

A developer writes an R script in SageMaker to train a logistic regression model on a 20 GB dataset in S3, with columns "age", "income", and "target". The script uses glm() and must handle missing values and scale features. Which snippet is correct?

A. library(aws.s3)df <- s3read_using(read.csv, bucket="bucket", object="data.csv")df[is.na(df)] <- 0df[, c("age", "income")] <- scale(df[, c("age", "income")])model <- glm(target ~ age + income, data=df, family=binomial)
B. library(boto3)
df <- read.csv("s3://bucket/data.csv")
df <- na.omit(df)
df$age <- (df$age - mean(df$age)) / sd(df$age)
df$income <- scale(df$income)
model <- glm(target ~ ., data=df, family="binomial")
C. library(data.table)
df <- fread("s3://bucket/data.csv")
df[is.na(df)] <- median(df, na.rm=TRUE)
df[, c("age", "income")] <- lapply(df[, c("age", "income")], scale)
model <- glm(target ~ age + income, family=binomial(link="logit"))

D. library(aws.s3)
df <- s3get_object(bucket="bucket", key="data.csv")
df <- impute(df, method="mean")
df[, c("age", "income")] <- normalize(df[, c("age", "income")])
model <- logist(target ~ age + income, data=df)

Answer: A

Explanation: The correct R script uses aws.s3's s3read_using to read from S3, replaces NAs with 0, scales features with scale(), and fits a binomial GLM. Option B uses Python's boto3, Option C misuses median and GLM syntax, and Option D has invalid functions (impute, normalize, logist).

Question: 896

A financial institution is building a fraud detection system using machine learning and has decided to use Amazon S3 as the primary storage medium for its datasets, which include transactional records and customer profiles. The data engineering team needs to ensure that the S3 bucket can handle a growing volume of data—currently 50 TB and expected to double annually—while supporting concurrent read/write operations from multiple SageMaker training jobs. Which configuration optimizes the S3 bucket for this ML use case?

A. Enable S3 versioning and configure lifecycle policies to transition older data to S3 Glacier, using default S3 storage class
B. Create an S3 bucket with Requester Pays enabled and use S3 Standard-Infrequent Access for all objects
C. Set up S3 bucket with Transfer Acceleration and multipart upload enabled, using S3 Intelligent-Tiering for cost optimization
D. Configure S3 bucket with cross-region replication to an EFS file system and enable strong consistency

Answer: C

Explanation: For a fraud detection ML system with large, growing datasets and concurrent SageMaker access, S3 must be optimized for performance and cost. Transfer Acceleration and multipart upload enhance upload speed and handle large files efficiently, while Intelligent-Tiering automatically adjusts storage costs based on access patterns. Versioning with Glacier is less optimal for frequent access, Requester Pays shifts costs inappropriately for internal use, and cross-region replication to EFS is impractical as EFS is a separate service, not an S3 feature.

Question: 897
In a SageMaker training job, you optimize a neural network with a custom loss function combining L1 and L2 penalties. The dataset has 5 million rows, and you use mini-batch gradient descent with a batch

size of 128 and learning rate of 0.005. After 40 epochs, the loss converges to 0.4 on training data but fluctuates between 0.7 and 0.9 on validation data. What's the most likely cause, and how should you address it?

A. Loss mismatch; switch to pure L2 loss
B. Learning rate too low; increase it to 0.01
C. Batch size too small; increase it to 512
D. Overfitting; add dropout with rate=0.3 to hidden layers

Answer: D

Explanation: Fluctuating validation loss with converged training loss indicates overfitting, where the model memorizes training data. Adding dropout (rate=0.3) regularizes the network, reducing overfitting and stabilizing validation performance without altering the optimization process.

Question: 898

A manufacturing firm is preparing a 26 TB dataset of production logs in S3 (Parquet format) for an ML model to predict quality. The dataset includes defect rates, pressures, and timestamps over 5 years. The team must create a histogram of defect rates to assess distribution, interpret the p-value from a t-test comparing pressures across shifts, and perform cluster analysis with an elbow plot to optimize cluster size (targeting 2-4 clusters). Which approach best analyzes and visualizes this data?

A. Set up an AWS Lambda function: plot a histogram with matplotlib.hist(), compute p-value with a hardcoded formula, and approximate clustering with a static size
B. Use AWS Glue with PySpark: create a histogram with pyplot.hist(), calculate p-value with a custom UDF, and perform hierarchical clustering with linkage() and an elbow plot from scipy
C. Configure Amazon QuickSight: build a histogram visual, estimate p-value manually, and skip clustering due to limited functionality
D. Deploy Amazon SageMaker with Jupyter Notebook: generate a histogram with seaborn.histplot(), compute p-value with scipy.stats.ttest_ind(), and use KMeans with elbow_plot() from sklearn

Answer: D

Explanation: Amazon SageMaker with Jupyter Notebook handles a 26 TB dataset efficiently: seaborn.histplot() visualizes defect rate distribution, ttest_ind() computes a precise p-value, and KMeans with an elbow plot optimizes cluster size. Glue lacks native statistical tools, QuickSight skips clustering, and Lambda is unsuitable for complex analysis.

Question: 899
You deploy a SageMaker model for real-time fraud detection using a gradient boosting classifier trained

on 5 million transactions. The model runs on an ml.m5.xlarge instance, and you need to update it every 15 minutes with 10,000 new transactions. Which online retraining strategy would minimize downtime?

A. Use a shadow endpoint with incremental updates via xgboost.train()
B. Retrain in batch mode every 15 minutes on an ml.p3.2xlarge instance
C. Implement a SageMaker endpoint with online SGD updates
D. Use a Lambda function to trigger full retraining

Answer: A

Explanation: A shadow endpoint with incremental updates via xgboost.train() allows seamless model updates without downtime, testing the new model in parallel before promotion. This ensures real-time availability while incorporating new data efficiently.

Question: 900

A company deploys a SageMaker endpoint with a PyTorch model on an ml.m5.large instance, handling 200 requests/minute. They need to add A/B testing for a new model version with 10% traffic. How should they configure this?

A. Deploy a second endpoint, use Application Load Balancer to split 10% traffic, and monitor with CloudWatch
B. Create a new endpoint variant with the new model, set its weight to 0.1, and update the existing endpoint
C. Use SageMaker Shadow Mode, deploy the new model as a shadow variant, and allocate 10% traffic
D. Configure SageMaker Multi-Model Endpoint, add the new model, and route 10% requests via inference logic

Answer: B

Explanation: SageMaker endpoint variants allow A/B testing by assigning weights (e.g., 0.1 for 10% traffic) to the new model within the same endpoint, simplifying management. ALB requires separate endpoints, Shadow Mode is for testing without live traffic, and Multi-Model Endpoints don’t support traffic splitting natively.

Question: 901

A manufacturing firm is processing a 18 TB dataset of sensor logs in S3 (CSV format), including temperatures, pressures, and failure flags, to minimize equipment downtime. The goal is to predict failure probability per machine with 92% accuracy, currently at 65% with manual checks. The ML team must decide if ML is appropriate, choose supervised vs. unsupervised learning, and select a model type,

considering labeled failure data. Which solution best frames this business problem?

A. Avoid ML: implement an AWS Glue job to flag machines above temperature thresholds, as ML is too complex
B. Frame as an unsupervised recommendation problem: use SageMaker with Factorization Machines to suggest maintenance schedules without failure predictions
C. Frame as a supervised classification problem: use SageMaker with XGBoost to predict failure probability, training on failure flags
D. Frame as a supervised regression problem: use SageMaker with Linear Learner to predict failure times as continuous values

Answer: C

Explanation: Predicting failure probability with 92% accuracy justifies ML over 65% manual checks. Supervised learning leverages labeled failure flags, and classification (XGBoost) suits the probabilistic outcome. Recommendation lacks failure focus, rule-based flagging underperforms, and regression misaligns with the binary prediction needed.

Question: 902

A gaming company is processing a 32 TB dataset of player logs in S3 (JSON format), including scores, playtimes, and churn flags, to reduce churn by 20%. The business aims to predict churn probability per player with 85% accuracy, currently at 55% with heuristic rules. The ML team must evaluate ML applicability, select supervised vs. unsupervised learning, and choose a model type, considering labeled churn data. Which solution best frames this business problem?

A. Frame as a supervised classification problem: use SageMaker with XGBoost to predict churn probability, training on churn flags
B. Frame as an unsupervised clustering problem: use SageMaker with K-Means to group players by playtimes, then analyze churn patterns
C. Avoid ML: implement an AWS Lambda function with playtime-based churn thresholds, as ML requires excessive tuning
D. Frame as a supervised regression problem: use SageMaker with Linear Learner to predict churn times as continuous values

Answer: A

Explanation: Predicting churn probability with 85% accuracy warrants ML over 55% heuristic rules. Supervised learning fits the labeled churn flags, and classification (XGBoost) addresses the probabilistic outcome. Clustering lacks predictive precision, rule-based thresholds underperform, and regression misaligns with the binary prediction needed.

Question: 903

You're training an RNN with 30 GB of time-series data using AWS Batch and Spot Instances on 5 p3.2xlarge instances. The job fails after 3 hours. How do you fix it?

A. Use On-Demand p3.2xlarge with no checkpointing and a 10-hour timeout
B. Add checkpointing to S3 every 10 epochs, set retries to 3, and use a 15-hour timeout
C. Switch to g4dn.xlarge Spot Instances with no retries
D. Run on SageMaker with Spot Instances and EBS storage

Answer: B

Explanation: Checkpointing to S3 every 10 epochs and retries handle Spot interruptions, ensuring completion on p3.2xlarge within 15 hours. On-Demand is costlier, and g4dn lacks GPU capacity.

Question: 904

A smart city initiative is implementing an ML model for traffic optimization and needs to transform streaming traffic camera data (200 MB/second) from Kinesis Data Streams. The transformation must occur in transit, aggregating vehicle counts by lane every 15 seconds, filtering out invalid frames, and saving as ORC in S3 partitioned by date (yyyy/MM/dd). Which solution best implements this data transformation in transit?

A. Set up Amazon Kinesis Data Firehose with a Lambda function to aggregate and filter data, writing to S3 without partitioning
B. Deploy Amazon EMR with Apache Spark Streaming, a 15-second micro-batch, and a custom job to aggregate, filter, and write ORC to S3
C. Configure AWS Batch with a Docker container running Apache Spark to process Kinesis data in 15-second batches and save ORC to S3
D. Use AWS Glue with a streaming ETL job, a PySpark script to aggregate by lane and filter invalid frames, and output to S3 with dynamic partitioning

Answer: D

Explanation: AWS Glue's streaming ETL with PySpark transforms Kinesis data in transit, aggregating by lane, filtering invalid frames, and partitioning ORC output to S3. EMR with Spark Streaming is complex, AWS Batch with Spark lacks streaming support, and Firehose with Lambda doesn't support advanced partitioning.

Question: 905
Which compute resource would be the most suitable for training a large-scale deep learning model that requires high computational power and parallel processing?

A. Standard CPU
B. High Memory Instance
C. GPU Instance
D. Low-Cost T2 Instance

Answer: C

Explanation: GPU instances are specifically optimized for high computational power and parallel processing, making them ideal for training large-scale deep learning models.

Question: 906

An energy analytics firm is implementing an ML model for demand forecasting and needs to orchestrate a pipeline that ingests real-time meter data (200 MB/second) into S3 and processes monthly batch data (12 TB, Parquet) from S3 with trend analysis. The streaming pipeline requires a 2-second latency, and the batch pipeline must run on the 1st of each month. Which services best orchestrate this hybrid pipeline?

A. Configure Amazon Kinesis Data Streams with 40 shards and Amazon Data Firehose for batch processing, orchestrated by Lambda
B. Deploy Amazon Managed Service for Apache Flink for streaming with a 2-second window and Amazon EMR for batch processing, managed by Step Functions
C. Use Amazon Kinesis Data Firehose for streaming to S3 with a 2-second buffer and AWS Glue for batch ETL with trend analysis, triggered by CloudWatch Events
D. Set up Amazon EMR with Spark Streaming for real-time ingestion and AWS Glue for batch processing, triggered by Data Pipeline

Answer: C

Explanation: Kinesis Data Firehose ingests streaming meter data (200 MB/s) into S3 with a 2-second buffer, while AWS Glue processes batch Parquet data monthly with trend analysis, orchestrated by CloudWatch Events. Managed Flink with EMR is complex, Kinesis Streams with Firehose misaligns roles, and EMR with Glue lacks streaming efficiency.

Question: 907
You deploy a k-means model in SageMaker to cluster IoT sensor data with 20 features, setting k=8 and using Euclidean distance. After clustering, you notice that one cluster contains 80% of the data points. What is the most likely issue, and how should you resolve it?

A. Uneven cluster sizes; switch to DBSCAN with eps=0.5
B. Wrong k; use the elbow method to find optimal k

C. Features on different scales; normalize data to [0, 1]
D. Outliers; remove points beyond 2 standard deviations

Answer: C

Explanation: K-means uses Euclidean distance, which is sensitive to feature scales. Unnormalized features can dominate the distance metric, causing imbalanced clusters. Normalizing data to [0, 1] ensures equal contribution from all features, improving cluster balance. Adjusting k or switching algorithms may help but doesn't address the scaling issue directly.

Question: 908

A large e-commerce company is designing a system to ingest real-time customer clickstream data from millions of users across multiple regions. The data, which includes user IDs, timestamps, product IDs, and session durations, must be collected at scale and stored in a data lake on Amazon S3 for downstream machine learning tasks. The ingestion pipeline must handle bursts of up to 10 GB/s, ensure low latency, and provide fault tolerance. Which combination of AWS services and configurations would best meet these requirements while minimizing operational overhead?

A. Set up Amazon SQS with a FIFO queue, process messages with an Auto Scaling group of EC2 instances, and upload data to S3 in Parquet format using the AWS SDK
B. Deploy an Amazon MSK (Managed Streaming for Kafka) cluster with 10 partitions, configure a custom consumer to batch data, and use AWS Lambda to write to S3 every 5 minutes
C. Use Amazon Kinesis Data Streams with 50 shards, enable enhanced fan-out, and write data directly to S3 using Kinesis Data Firehose with a buffer interval of 60 seconds
D. Use Amazon API Gateway with a WebSocket connection to ingest data, process it with AWS AppSync, and store it in S3 via a GraphQL mutation every 10 seconds

Answer: C

Explanation: For high-throughput, real-time ingestion at 10 GB/s with low latency and fault tolerance, Amazon Kinesis Data Streams is ideal due to its scalability and ability to handle massive data streams. With 50 shards (each supporting 1 MB/s ingress), it can manage the load, and enhanced fan-out ensures low-latency delivery to consumers. Kinesis Data Firehose seamlessly integrates with S3, buffering data (e.g., 60 seconds) to optimize writes, reducing operational complexity compared to custom solutions. MSK is powerful but requires more management for consumers, SQS isn't suited for such high throughput, and API Gateway with WebSocket is impractical for this scale of raw data ingestion.

Question: 909

A pharmaceutical company is building an ML model for drug discovery and needs to ingest streaming

sensor data (90 MB/second) from lab equipment into an S3 data lake. The ingestion must aggregate data by experiment ID every 30 seconds, partition by date and equipment ID (yyyy/MM/dd/equipID), and handle late-arriving events up to 2 minutes. Which streaming ingestion solution is most appropriate?

A. Use Amazon Managed Service for Apache Flink with a 30-second tumbling window, late event handling (2 minutes), and a partitioned S3 sink
B. Configure Amazon Kinesis Data Firehose with a 30-second buffer and a Lambda function for aggregation and partitioning, with no late event support
C. Deploy Amazon EMR with Apache Spark Streaming, a 30-second micro-batch, and a custom script to aggregate and partition to S3
D. Set up Amazon Kinesis Data Streams with 18 shards and a consumer Lambda to aggregate and partition data to S3 every 30 seconds

Answer: A

Explanation: Managed Service for Apache Flink excels at streaming with a 30-second tumbling window, late event handling (2 minutes), and custom S3 sinks with partitioning (date/equipID). Firehose lacks late event support, EMR with Spark is batch-heavy, and Kinesis Streams with Lambda requires more custom logic.

Question: 910

A media company trains a SageMaker model to classify video content as "viral" or "non-viral" using 10,000 samples (20% viral). The confusion matrix on a test set is: TP = 1,500, FP = 500, TN = 7,500, FN = 500. What is the recall, and what does it imply for the model’s performance?

A. 0.60, showing moderate success in predicting viral videos
B. 0.83, suggesting high reliability in detecting non-viral content
C. 0.75, indicating 75% of viral videos are correctly identified
D. 0.90, reflecting strong overall classification performance

Answer: C

Explanation: Recall = TP / (TP + FN) = 1,500 / (1,500 + 500) = 0.75. This means 75% of actual viral videos are correctly classified, implying the model is reasonably effective at identifying viral content but misses 25% of viral cases, which could be critical depending on the business use case.

Question: 911

A data science team trains an ML model on SageMaker with a 1 TB dataset, requiring persistent block storage with snapshots for rollback (e.g., volume size 1024 GiB, IOPS 3000). The storage must attach to ml.c5.xlarge instances and encrypt data at rest. What should they use?

A. Deploy Amazon EFS with SageMaker integration
B. Use Amazon EBS with gp3 volumes and encryption
C. Configure Amazon FSx with block storage
D. Set up Amazon S3 with lifecycle policies

Answer: B

Explanation: Amazon EBS gp3 volumes provide persistent block storage (e.g., 1024 GiB, 3000 IOPS) with snapshots and encryption, attaching to SageMaker instances like ml.c5.xlarge. EFS is file-based, FSx is for specific file protocols, and S3 is object storage, none of which offer block-level persistence.

Question: 912

You are tasked with deploying a new model version using Amazon SageMaker and need to ensure minimal disruption to your users while switching from the old model. What deployment strategy should you consider?

A. Canary deployment
B. Rolling update
C. Blue/Green deployment
D. All-at-once deployment

Answer: C

Explanation: Blue/Green deployment allows for seamless switching between the old and new model versions, minimizing user disruption and allowing for easy rollback if issues arise.

Question: 913

A telecom provider is analyzing a 17 TB dataset of signal logs in S3 (JSON format) for an ML model to predict quality. The dataset includes strengths, latencies, and timestamps over 4 years. The team needs to create a scatter plot of strength vs. latency, calculate the Pearson correlation between these variables, and perform hierarchical clustering with a dendrogram to diagnose network segments. Which solution best accomplishes this visualization and analysis?

A. Configure Amazon QuickSight: build a scatter plot visual, estimate correlation manually, and skip clustering due to lack of support
B. Deploy Amazon SageMaker with Jupyter Notebook: create a scatter plot with seaborn.scatterplot(), calculate correlation with pandas.corr(), and use KMeans with a static cluster size
C. Use AWS Glue with PySpark: generate a scatter plot with pyplot.scatter(), compute correlation with corr(), and perform hierarchical clustering with scipy.cluster.hierarchy.dendrogram()

D. Set up an AWS Lambda function: plot a scatter with matplotlib.scatter(), compute correlation with a custom formula, and approximate clustering without visualization

Answer: C

Explanation: AWS Glue with PySpark scales for a 17 TB dataset: pyplot.scatter() visualizes strength vs. latency, corr() computes Pearson correlation, and dendrogram() from scipy diagnoses hierarchical clustering. SageMaker lacks hierarchical clustering, QuickSight skips clustering, and Lambda is impractical for large-scale visualization.

Question: 914
In the context of model evaluation, what does a high AUC-ROC value signify regarding the model's ability to classify positive and negative instances?

A. The model has poor classification accuracy.
B. The model's predictions are unreliable.
C. The model is not overfitting.
D. The model performs well, effectively distinguishing between positive and negative instances.

Answer: D

Explanation: A high AUC-ROC value indicates that the model performs well in distinguishing between positive and negative instances, reflecting strong classification capabilities.

Question: 915
A medical imaging dataset has inconsistent lighting across X-rays. Which preprocessing step standardizes the images?

A. Use AWS Rekognition to auto-tag images
B. Convert images to grayscale and crop edges
C. Rescale pixel values to [0, 1] and apply histogram equalization
D. Delete underexposed/overexposed images

Answer: C

Explanation: Histogram equalization normalizes contrast, and rescaling ensures consistent input ranges. Grayscale conversion alone doesn’t fix lighting, and deleting images reduces dataset size unnecessarily.

Question: 916

What is the most common reason for a model to converge to a local minimum instead of a global minimum during training?

A. The choice of optimization algorithm.
B. The complexity of the dataset.
C. The size of the training data.
D. The presence of non-convex loss functions.

Answer: D

Explanation: Non-convex loss functions can lead to multiple local minima, causing the optimization algorithm to converge to a local minimum rather than the global minimum.

KILLEXAMS.COM

Killexams.com is a leading online platform specializing in high-quality certification exam preparation. Offering a robust suite of tools, including MCQs, practice tests, and advanced test engines, Killexams.com empowers candidates to excel in their certification exams. Discover the key features that make Killexams.com the go-to choice for exam success.

Exam Questions:

Killexams.com provides exam questions that are experienced in test centers. These questions are updated regularly to ensure they are up-to-date and relevant to the latest exam syllabus. By studying these questions, candidates can familiarize themselves with the content and format of the real exam.

Exam MCQs:

Killexams.com offers exam MCQs in PDF format. These questions contain a comprehensive collection of questions and answers that cover the exam topics. By using these MCQs, candidate can enhance their knowledge and improve their chances of success in the certification exam.

Practice Test:

Killexams.com provides practice test through their desktop test engine and online test engine. These practice tests simulate the real exam environment and help candidates assess their readiness for the actual exam. The practice test cover a wide range of questions and enable candidates to identify their strengths and weaknesses.

Guaranteed Success:

Killexams.com offers a success guarantee with the exam MCQs. Killexams claim that by using this materials, candidates will pass their exams on the first attempt or they will get refund for the purchase price. This guarantee provides assurance and confidence to individuals preparing for certification exam.

Updated Contents:

Killexams.com regularly updates its question bank of MCQs to ensure that they are current and reflect the latest changes in the exam syllabus. This helps candidates stay up-to-date with the exam content and increases their chances of success.