databricks

Databricks 試験ガイド

Databricks 認定Generative AI エンジニアアソシエイト

試験ガイドに関するフィードバックを送信する

この試験ガイドの目的

この試験ガイドでは、試験の概要と出題範囲を解説し、受験準備の進捗を確認できるよう支援します。本資料は、試験内容に変更があった場合（およびその変更が試験に適用される時期）に随時更新されますので、十分な準備が可能です。本版は、**2026年3月18日**時点の現行試験内容を反映しています。受験の**2**週間前には必ず最新版を確認してください。

対象者についての説明

“Databricks 認定Generative AI エンジニアアソシエイト” 認定試験では、Databricks を利用して LLM 対応ソリューションを設計および実装する個人の能力を評価します。具体的には、複雑な要件を管理可能なタスクに分解する問題分解能力や、包括的なソリューション開発のために、現在のGenerative AI 環境から適切なモデル、ツール、アプローチを選択する能力を含みます。また、セマンティック類似性検索用の Vector Search、モデルおよびソリューションのデプロイ用の Model Serving、ソリューションライフサイクル管理用の MLflow、データガバナンス用の Unity Catalog などの、Databricks 固有のツールに関する知識も評価されます。本試験に合格された方は、Databricks とそのツールセットを最大限に活用した、高性能な RAG アプリケーションおよび LLM チェインの構築とデプロイを遂行できることを期待されます。

試験について

- 採点対象問題数：45 問（単一選択または複数選択）*
- 試験時間：90 分
- 受験料: $200
- 実施方法: オンライン監督付き
- 前提条件： 特にありません。関連するコースの受講および 6 ヶ月の実務経験を推奨しています。
- 有効期間：2 年間
- 再認定：認定資格の維持には、2 年ごとの再認定が必要です。再認定を受けるためには、現在有効な試験を受検する必要があります。試験ウェブページの「試験準備」セクションを確認し、再度の受験に備えてください。
- 採点対象外の問題：本試験には、将来的な統計情報の収集のため、採点対象外の問題が含まれる場合があります。これらは試験の画面では特定されず、スコアには影響しません。

採点対象外の問題を含めるため、試験時間には追加の時間枠が設定されています。

推奨される準備

- Generative AI学習者の役割に関連する現在のDatabricks Academy ILTコース、特にDatabricksを使用したGenerative AIエンジニアリング。
- 自分のペースで学べる（Databricks Academyで提供）：Databricksを活用したGenerative AIエンジニアリング。以下のコースが含まれます：
    - Databricks でのリトリーブエージェントの構築
    - Databricks でのシングルエージェントアプリケーションの構築
    - Generative AIアプリケーションの評価とガバナンス
    - Generative AIアプリケーションのデプロイと監視
- 最新の LLM とその機能に関する知識
- プロンプトエンジニアリング、プロンプト生成、評価に関する知識
- LangChain、Hugging Face Transformersなどの現在の関連するオンラインツールやサービスに関する知識。
- PythonおよびRAGアプリケーション、エージェント、LLMチェーンの開発をサポートするライブラリに関する実務知識
- データプレパレーション、モデルチェーニングなどに関する最新のAPIの実務知識
- 関連する Databricks ドキュメントリソース

試験の概要

セクション **1:** アプリケーションの設計

- 特定の形式で応答を引き出すプロンプトを設計する
- 所定のビジネス要件を満たすモデルタスクを選択する
- 希望するモデルの入力および出力に合わせてチェーンコンポーネントを選択する
- ビジネスユースケースの目標を、AIパイプラインの希望する入力および出力の記述に変換する
- 多段階推論のために知識を収集したりアクションを実行したりするツールを定義し、順序付けする
- 問題解決のためにAgent Bricks（ナレッジアシスタント、マルチエージェントスーパーバイザー、情報抽出）をいつ、どのように使用するかを決定する

セクション **2:** データの準備

- 指定された文書構造とモデルの制約条件に合わせてチャンキング戦略を適用する
- RAGアプリケーションの品質を低下させるソース文書内の不要なコンテンツをフィルタリングする
- 提供されたソースデータとフォーマットから文書コンテンツを抽出するために、適切なPythonパッケージを選択する
- 指定されたチャンク化されたテキストをUnity Catalog内のDelta Lakeテーブルに書き込むための操作と順序を定義する
- 特定のRAGアプリケーションに必要な知識と品質を提供するソース文書を特定する
- ツールや指標を使用して検索パフォーマンスを評価する
- 高度なチャンキング戦略を用いて検索システムを設計する
- 情報検索プロセスにおける再ランク付けの役割を説明する

セクション **3:** アプリケーション開発

- Generative AIアプリケーションで使用するLangchainや類似のツールを選定する。
- 応答を定性的に評価し、品質や安全性などの一般的な課題を特定する
- モデルおよび検索評価に基づいてチャンキング戦略を選定する
- キーフィールド、用語、意図に基づいて、ユーザー入力からの追加コンテキストでプロンプトを拡張する
- LLMの応答を、ベースラインから望ましい出力へと調整するプロンプトを作成する
- 望ましくない結果を防止するためのLLMガードレールを実装する
- 開発するアプリケーションの属性に基づいて最適なLLMを選択する
- ソース文書、想定されるクエリ、最適化戦略に基づいて、埋め込みモデルのコンテキスト長を選択する
- モデルのメタデータやモデルカードに基づいて、タスクに適したモデルをモデルハブやMarketplaceから選択する
- 実験で生成された一般的な指標に基づき、特定のタスクに最適なモデルを選択する
- MLflowおよびAgent Frameworkを活用してagentic systems を開発する
- Gen AIアプリケーションのライフサイクルにおける評価フェーズとモニターリングフェーズを比較する
- Genie Spacesまたは会話型APIを活用してデータを取得するマルチエージェントシステムを実現する

セクション **4:** アプリケーションの組み立てとデプロイ

- 前処理および後処理を含む pyfuncs モデルを使用したチェーンを実装する
- model serving endpointsからのリソースへのアクセスを制御する
- 要件に基づいてシンプルなチェーンを実装する
- RAG アプリケーションの作成に必要な基本要素（モデルフレーバー、埋め込みモデル、リトリーバー、依存関係、入力例、モデルシグネチャ）を選択する
- MLflow を使用して Unity Catalog にモデルを登録する
- Vector Searchインデックスを作成し、クエリを実行する
- ファウンデーションモデルAPIを活用したLLMアプリケーションの提供方法を特定する
- Mosaic AI Vector Searchの主要な概念とコンポーネントを説明する
- バッチ推論ワークロードを特定し、ai_query()を適切に適用する
- 埋め込み数、更新頻度、レイテンシ、コスト要件に基づいて、特定のソリューション向けにVector Searchを構成する
- 中間メモリや構造化情報を保存・取得するための永続データストアを構成する
- Vector Searchインデックスの更新、環境間でのプロンプトの展開、エージェントの個々のコンポーネントのテストなど、CI/CDのベストプラクティスを適用する
- 特定のアプリケーション要件に基づき、マネージド、外部、およびカスタムMCPサーバーを統合する
- プロンプトのバージョン管理を適用し、プロンプトのライフサイクルを管理する
- エージェントの使用シナリオ（アプリ、Slack、Teamsなど）に適した対話型ユーザーインターフェースを開発する

セクション **5:** ガバナンス

- パフォーマンス目標を達成するためのガードレールとして、マスキング手法を活用する
- Gen AIアプリケーションへの悪意のあるユーザー入力から保護するためのガードレール手法を選択する
- 法的リスクを回避するために、データソースに関する法的要件やライセンス要件を活用する
- Gen AIアプリケーションにデータを提供するデータソース内の問題のあるテキストを軽減するため

の代替策を提案する

セクション 6: 評価とモニタリング

- 一連の定量的評価指標に基づいて、LLM（サイズとアーキテクチャ）を選択する
- 特定のLLMデプロイメントシナリオにおいて監視すべき主要な指標を選択する
- MLflowのスコアリングとトレース機能を使用してエージェントのパフォーマンスを評価する
- 推論ログを使用して、デプロイされたRAGアプリケーションのパフォーマンスを評価する
- Databricksの機能を使用してLLMのコストを管理する
- 推論テーブルとエージェントモニタリングを使用して、稼働中のLLM endpoint を追跡する
- グラウンドトゥルースを必要とする評価ジャッジを特定する
- AI Gateway（推論テーブル、使用状況テーブル、レート制限）を使用して、エージェントframework 経由でデプロイされたLLMまたはエージェントを追跡する
- Databricksのカスタムスコアラーを使用して、エージェントとLLMを評価する
- 専門家のフィードバックを取り入れて、エージェントのパフォーマンスを改善する

サンプル質問

これらの問題は実際の試験問題と類似しており、本試験でどのような形式で出題されるか、その概要を把握するのに役立ちます。各問題には、試験ガイドに記載されている試験の学習目標が含まれており、その目標に沿ったサンプル問題も掲載されています。試験ガイドには、試験で出題される可能性のあるすべての学習目標が記載されています。認定試験の準備をする上で最も効果的な方法は、試験ガイドに記載されている試験の概要を確認することです。

問題 **1**

目的：与えられた文書構造とモデルの制約条件に対して、チャンキング戦略を適用する

あるGenerative AIエンジニアが、最大1億件までしか格納できないvectorデータベースに、1億5000万件の埋め込みデータをロードしようとしている。

レコード数を減らすために実行できる 2 つのアクションはどれですか。

A. ドキュメントのチャンク サイズを大きくする
B. チャンク間のオーバーラップを減らす
C. ドキュメントのチャンク サイズを小さくする
D. チャンク間のオーバーラップを増やす
E. より小さな埋め込みモデルを使用する

問題 **2**

目的：特定のRAGアプリケーションに必要な知識と品質を提供する、必要なソースドキュメントを特定する。

あるGenerative AIエンジニアは、自動車部品の販売支援を目的として開発中の顧客向けGen AIアプリケーションからの応答を評価している。このアプリケーションでは、質問に回答するために顧客が *account_id* と *transaction_id* を明示的に入力する必要がある。初期リリース後、顧客からのフィードバックによると、注文や請求の詳細に関する回答は良好であったものの、配送や到着予定日に関する質問には正確に回答できていなかった。

以下のうち、これらの質問に対するアプリケーションの回答能力を向上させるアプローチはどれか?

A. すべての自動車部品に関する会社の配送ポリシーと支払条件を含むvector storeを作成する
B. 取引IDをプライマリキーとすること、請求書データと配送予定日が格納されるfeature storeテーブルを作成する

C. 配送予定日のサンプルデータをチューニング用データセットとして提供し、配送情報が最新のものになるよう、定期的にモデルの微調整を行う
D. 注文日時を入力するチャットプロンプトを修正し、配送方法が14日を超えることはないと想定されるため、その日付に14日を加算するようモデルに指示する

問題 **3**

目的：提供されたソースデータとフォーマットからドキュメントの内容を抽出するために、適切な*Python*パッケージを選択する。

あるGenerative AIエンジニアが、スキャンされて.jpegや.pngなどのフォーマットで画像ファイルとして保存されたソース文書から取得したコンテキストを利用するRAGアプリケーションを構築しています。彼らは、最小限のコード行数でソリューションを開発したいと考えています。

ソース文書からテキストを抽出するには、どのPythonパッケージを使用すべきですか？

A. beautifulsoup
B. scrapy
C. pytesseract
D. pyquery

問題 **4**

目的：ソース文書、想定されるクエリ、および最適化戦略に基づいて、埋め込みモデルのコンテキスト長を選択する

あるGenerative AIエンジニアが、LLMベースのアプリケーションを開発している。リトリバー用の文書は、それぞれ最大512トークンに分割されている。Generative AI エンジニアは、このアプリケーションにおいて品質よりもコストとレイテンシの方が重要であることを認識している。選択可能なコンテキスト長にはいくつかのレベルがある。

どの選択肢が要件を満たすか？

A. コンテキスト長 512：最小モデルは 0.13GB、埋め込みディメンション 384
B. コンテキスト長 514：最小モデルは 0.44GB、埋め込みディメンション 768
C. コンテキスト長 2048：最小モデルは 11GB、埋め込みディメンション 2560
D. コンテキスト長 32768：最小モデルは14GB、埋め込みディメンション4096

問題 **5**

目的：開発するアプリケーションの特性に基づいて、最適な*LLM*を選択する

あるGenerative AIエンジニアは、1段落程度の長さがあるメモフィールドの内容を、その意図を伝えつつ、アプリケーションのフロントエンドに収まる1文の要約に更新できるアプリケーションを構築したいと考えている。

このアプリケーション向けに候補となるLLMを評価する際、どの自然言語処理タスクカテゴリを用いるべきか？

A. text2text ジェネレーション
B. センテナイザー
C. テキストクラシフィケーション
D. スサマライゼーション

問題 **6**

目的：埋め込み数、更新頻度、レイテンシ、およびコスト要件に基づいて、特定のソリューション向けに*vector search*を設定する。

あるオンライン小売業者に勤務するGenerative AIエンジニアが、vector searchとメタデータフィルタリングを組み合わせることで、検索機能の改善を試みている。検索数は1秒あたり最大80件に達し、レイテンシが最も重要な指標となっている。レイテンシを損なうことなく精度が向上するのであれば、初期の開発コストは問題としない。在庫は全国で1億点に上る。

エンジニアはどのように設定すべきか？

A. GTE Large埋め込みモデルを活用し、ハイブリッド検索と再ランク付けを有効にした標準vector searchを使用する。
B. GTE Large埋め込みモデルを活用し、ハイブリッド検索と再ランク付けを有効にしたストレージ最適化vector searchを使用する。
C. カスタム埋め込みモデルを微調整し、標準vector searchを使用し、ハイブリッド検索と再ランク付けは無効にする。
D. カスタム埋め込みモデルを微調整し、ストレージ最適化vector searchを使用し、ハイブリッド検索と再ランク付けは無効にする。

問題 **7**

目的：*Vector Search*のインデックス更新、環境間でのプロンプトの展開、エージェントの個々のコンポーネントのテストなど、*CI/CD*のベストプラクティスを適用する。

Generative AIエンジニアは、開発、ステージング、本番環境にわたるエージェントのプロンプトテンプレートを管理する必要がある。チームでは、段階的なリリースプロセスを採用している。具体的には、開発環境でプロンプトを更新し、ステージング環境で自動テストにより検証を行い、承認された後にのみ本番環境へ展開する。ソリューションはバージョン履歴を保持し、必要に応じて以前のプロンプトバージョンへロールバックできるものでなければなりません。

このプロモーションワークフローをサポートするアプローチはどれですか？

A. プロンプトテンプレートをアプリケーションリポジトリに保存し、テストに合格した後、ステージングブランチを本番ブランチにマージすることでプロモートする。
B. プロンプトをMLflowバージョンとして追跡し、テストに合格した後、エイリアスを使用してプロモートする。
C. CIランナー上のJSONファイルにプロンプトを保存し、実行のたびに本番環境のプロンプトを上書きする。
D. プロンプトをDeltaテーブルに格納し、一貫性を確保するためにデプロイのたびに本番環境のテーブルを上書きする。

問題 **8**

目的：エージェント利用シナリオ（アプリ、*Slack*、*Teams*など）に適した対話型ユーザーインターフェースを開発する。

Generative AIエンジニアは、カスタマーサポートのエージェントが質問を行い、社内のPDFに基づいた回答を受け取れるDatabricks Appを構築している。要件：ユーザーは企業IDで認証を行う必要があり、アプリはブラウザ内で長期有効なトークンを公開することなくMosaic AI Agent endpointを呼び出し、回答へのアクセスは各ユーザーの権限を遵守する必要があります。

これらの要件を満たすアプローチはどれですか？

A. Databricks Appのバックエンドを使用して、アプリの認証情報でAgent endpointを呼び出し、アプリの認証済みコンテキストを通じてユーザーのIDと権限を適用する。
B. Databricksのpersonal access token（PAT）をアプリのJavaScriptに保存し、ブラウザから直接エージェントendpointを呼び出す。
C. エージェントendpointを公開し、アプリのフロントエンドに埋め込まれたAPIキーで保護する。
D. PDFをパブリックバケットにエクスポートし、エージェントがID確認なしで読み込めるようにする。

問題 **9**

目的：与えられたアプリケーション要件に基づき、マネージド、外部、およびカスタムの*MCP*サーバーを統合する。

あるGenerative AIエンジニアが、インターネットのデータソースから事実情報を取得し、外部APIを使用してウェブ検索を実行する必要があるリサーチアシスタントエージェントを構築しています。Databricksはこのインターネットデータソース用にマネージドMCPサーバーを提供しており、キーを必要とする外部APIには外部MCPサーバーが利用可能です。アプリケーションは、本番環境において両方のデータソースへの信頼性の高いアクセスを確保しつつ、メンテナンスのオーバーヘッドを最小限に抑える必要があります。

これらのデータソースをエージェントに統合するために、エンジニアはどの2つのアクションを実行すべきですか？

A. インターネットリソースと外部APIの両方を、エージェントが呼び出せる単一の統合インターフェースにラップするカスタムMCPサーバーを構築する。
B. マネージドウェブブラウザMCPサーバーを使用して、情報を取得するためにプログラムでインターネットリソースへナビゲートする。
C. Unity Catalogの外部テーブルを設定し、インターネットリソースのコンテンツと検索結果をキャッシュして、エージェントがオフラインでアクセスできるようにする。
D. エージェントのMCPサーバー設定を通じて、サーバータイプを「マネージド」に指定し、インターネットリソースのサーバー識別子を提供することで、マネージドMCPサーバーを設定する。
E. 外部MCPサーバーの接続詳細を提供し、APIキーをDatabricks Secretsに保存し、MCPサーバー設定でそれを参照することで、外部MCPサーバーをデプロイする。

問題 **10**

目的：*SME*からのフィードバックを取り入れ、エージェントのパフォーマンスを向上させる。

あるGenerative AIエンジニアは、運用チームが社内で使用しているカスタマーサポート用RAGアシスタントの評価を担当している。4人のドメインエキスパートが毎週、MLflow上で抽出された回答を、事実の正確性、網羅性、有用性などの観点からレビューしている。数回の評価を経て、エンジニアは、同じ回答に対してエキスパートの評価に大きなばらつきがあることに気づいた。そのため、この評価データでは、モデルの経時的な改善状況を追跡することが困難になっている。エンジニアは、評価者間の不一致を軽減しつつ、反復的な品質改善を支援する、信頼性の高い評価プロセスを構築する必要があります。

エンジニアはどのように対応すべきでしょうか？

A. LLMをジャッジとして活用し、過去および将来の回答を再採点し、専門家の意見の相違を調整するのではなく、モデルが生成した評価を主要な信頼できる情報源として扱う。
B. 明確な評価基準（ルーブリック）を定義し、基準に関してSMEを校正し、整合されたジャッジメントを`mlflow.genai.evaluate()`で活用することで、一貫したエージェント評価を実現します。
C. 各回答についてすべての専門家のスコアを平均化し、その統合スコアをモデルチューニングの決定的なベンチマークとして直接使用する。
D. すべての専門家がすでに合意している回答のみからベンチマークを構築し、一貫性を高めるために評価セットから意見が分かれているケースを除外する。

解答

問題1：A、B

問題2：B

問題3：C

問題4：A

問題5：D

問題6：C

問題7：B

問題8：A

問題9：D、E

問題10：B