databricks

Databricks 시험 가이드

Databricks 인증 Machine Learning Associate

이 시험 가이드의 목적

이 시험 가이드는 시험 개요와 시험 내용에 대한 정보를 제공하여 시험 준비 상태를 판단하는 데 도움이 됩니다. 이 문서는 시험에 변경 사항이 있을 때마다(그리고 해당 변경 사항이 시험에 적용될 때) 업데이트되므로 그에 맞춰 준비할 수 있습니다. 이 버전은 **2024**년 **10**월 **28**일 현재 라이브 버전을 다룹니다**.** 시험을 치르기 **2**주 전에 다시 확인하여 최신 버전을 사용하고 있는지 확인하세요**.**

응시자 개요

Databricks Certified Machine Learning Associate 인증 시험은 개인이 Databricks를 사용하여 기본적인 머신 러닝 작업을 수행하는 능력을 평가합니다. Databricks와 Databricks의 머신 러닝 기능인 AutoML, Unity Catalog, MLflow의 일부 기능을 이해하고 사용하는 능력이 이에 포함됩니다. 또한 데이터를 탐색하고 특성 엔지니어링(Feature engineering)을 수행하는 능력도 평가합니다. 또한 시험에서는 훈련, 튜닝, 평가 및 선택을 통한 모델 구축을 평가합니다. 마지막으로, 머신 러닝 모델을 배포하는 능력이 평가됩니다. 이 인증 시험에 합격한 개인은 Databricks와 관련 도구를 사용하여 기본적인 머신 러닝 작업을 완료할 수 있습니다.

시험에 관하여

- 항목 수: 48개의 객관식 또는 복수 선택 문제
- 시간 제한: 90분
- 응시료: $200
- 시험 진행 방식: 온라인 감독
- 테스트 보조 도구: 허용되지 않음
- 필수 조건: 특별한 요구사항은 없습니다. 강의 수강 및 아래 시험 개요에 언급된 작업을 수행한 6개월의 실무 경험을 적극 권장합니다. 또한 이 문서의 권장 준비 사항을 참조하세요.
- 유효 기간: 2년
- 재인증: 인증 상태를 유지하려면 2년마다 재인증을 받아야 합니다. 재인증을 받으려면 현재 시행 중인 전체 시험에 응시해야 합니다. 시험에 다시 응시할 준비를 하려면 시험 웹페이지의 '시험 준비하기' 섹션을 검토하세요.
- 점수가 매겨지지 않은 콘텐츠: 시험에는 향후 사용을 위한 통계 정보를 수집하기 위해 점수가 매겨지지 않은 항목이 포함될 수 있습니다. 이러한 항목은 양식에 명시되지 않으며 점수에 영향을 미치지 않습니다. 이 콘텐츠에는 추가 시간이 계정에 반영됩니다.

추천 준비 과정

- 강사 주도: Databricks를 사용한 머신 러닝
- 자기 주도 학습(Databricks Academy에서 이용 가능): Databricks를 사용한 머신 러닝
- Python 및 scikit-learn, 스파크ML과 같은 머신 러닝을 지원하는 주요 라이브러리에 대한 실무 지식
- Unity Catalog 및 Delta Live Tables와 같은 기타 Databricks 데이터 관리 기능에 대한 실무 지식
- Databricks Documentation의 머신 러닝의 주요 주제에 대한 지식

시험 개요

섹션 **1: Databricks Machine Learning**

- MLOps 전략의 모범 사례를 파악합니다.
- ML 런타임을 사용하는 이점을 파악합니다.
- AutoML이 모델/특성 선택을 용이하게 하는 방법을 파악합니다.
- AutoML이 모델 개발 프로세스에 가져다주는 이점을 파악합니다.
- Databricks의 Unity Catalog에서 계정 수준에서 특성 저장소(feature store) 테이블을 만드는 이점을 워크스페이스 수준에서 만드는 경우와 비교하여 파악합니다.
- Unity Catalog에 특성 저장소(feature store) 테이블을 생성합니다.
- 특성 저장소 테이블에 데이터를 기록합니다.
- 특성 저장소 테이블의 특성(feature)를 사용하여 모델을 훈련합니다.
- 특성 저장소 테이블의 특성을 사용하여 모델의 점수를 매깁니다.
- 온라인 및 오프라인 특성 저장소 테이블의 차이점을 설명합니다.
- MLflow 클라이언트 API를 사용하여 가장 좋은 실행(Run)을 식별합니다.
- MLflow 실행(Run)에서 메트릭, 아티팩트 및 모델을 수동으로 기록합니다.
- MLFlow UI에서 사용 가능한 정보를 파악합니다.
- Unity Catalog 레지스트리에서 MLflow Client API를 사용하여 모델을 등록합니다.
- 워크스페이스 레지스트리보다 Unity Catalog 레지스트리에 모델을 등록했을 때의 이점을 파악합니다.
- 모델 승격(Promotion)보다 코드 승격(Promotion)이 더 선호되는 시나리오 및 반대의 경우 모두를 파악합니다.
- 모델에 태그를 설정하거나 제거합니다.
- 별칭(aliases)을 사용하여 챌린저 모델을 챔피언 모델로 승격합니다.

섹션 **2:** 데이터 처리

- Spark DataFrame에 대한 요약 통계를 `.summary()` 또는 dbutils 데이터 요약을 사용하여 계산합니다.
- 표준 편차 또는 IQR을 기반으로 Spark DataFrame에서 이상치를 제거합니다.
- 범주형 또는 연속형 특성에 대한 시각화를 만듭니다.
- 적절한 방법을 사용하여 두 개의 범주형 또는 두 개의 연속형 특성을 비교합니다.
- 평균, 중앙값 또는 최빈값으로 누락된 값을 대체하는 방법을 비교하고 대조합니다.

- 누락된 값을 최빈값, 평균 또는 중앙값으로 대체합니다.
- 범주형 특성에 대해 원-핫 인코딩(one-hot encoding)을 사용합니다.
- 원-핫 인코딩이 적절하거나 적절하지 않은 모델 유형이나 데이터 세트를 식별하고 설명합니다..
- 로그 스케일 변환이 적절한 시나리오를 파악합니다.

섹션 **3:** 모델 개발

- ML을 사용하여 주어진 모델 시나리오에 적합한 알고리즘을 선택합니다.
- 훈련 데이터의 데이터 불균형을 완화하는 방법을 식별합니다.
- 추정기(estimators)와 변환기(transformers)를 비교합니다.
- 훈련 파이프라인을 개발합니다.
- Hyperopt의 **`fmin`** 작업을 사용하여 모델의 하이퍼파라미터를 조정합니다.
- 하이퍼파라미터를 조정하는 방법으로 무작위 탐색, 그리드 탐색 또는 Bayesian 탐색을 수행합니다.
- 하이퍼파라미터 튜닝을 위한 단일 노드 모델을 병렬화합니다.
- 학습-검증 분할 대신 교차 검증을 사용하는 것의 장점과 단점을 설명합니다.
- 모델 피팅의 일부로 교차 검증을 수행합니다.
- 그리드 서치 및 교차 검증 과정을 함께 수행할 때 훈련되는 모델의 수를 파악합니다.
- 일반적인 분류 지표를 사용 : F1, Log Loss, ROC/AUC 등
- 일반적인 회귀 지표를 사용 : RMSE, MAE, R-squared 등
- 주어진 시나리오 목표에 가장 적합한 지표를 선택합니다.
- 평가 메트릭을 계산하거나 예측을 해석하기 전에 로그 변환된 변수를 지수화해야 할 필요성을 식별합니다.
- 모델 복잡성과 편향 분산 트레이드오프가 모델 성능에 미치는 영향을 평가합니다.

섹션 **4:** 모델 배포

- 모델 서빙 접근 방식의 차이점과 장점을 파악합니다: 배치, 실시간 및 스트리밍
- 사용자 정의 모델을 모델 엔드포인트에 배포합니다.
- pandas를 사용하여 배치 추론을 수행합니다,
- Delta Live Tables를 사용하여 스트리밍 추론이 수행되는 방법을 파악합니다.
- 실시간 추론을 위해 모델을 배포하고 쿼리합니다.
- 실시간 추론를 위해 엔드포인트 간에 데이터를 분할합니다.

질문 샘플

이러한 문제는 실제 문제와 유사하며 이 시험에서 문제가 어떻게 출제되는지를 일반적으로 이해할 수 있도록 제공된 것입니다. 샘플 문제에는 시험 가이드에 명시된 시험 목표가 포함되어 있으며, 목표에 맞는 샘플 문제가 제공됩니다. 시험 가이드에는 시험에서 다룰 수 있는 모든 목표가 나열되어 있습니다. 인증 시험을 준비하는 가장 좋은 방법은 시험 가이드의 시험 개요를 검토하는 것입니다.

질문 **1**

목표: *Unity Catalog*에 특성 테이블*(feature table)*을 만듭니다.

데이터 과학자는 모델에서 사용할 특성 테이블을 만들자 합니다. 그들은 Unity Catalog가 활성화된 워크스페이스에서 작업하고 있으며 이 특성 테이블을 저장하고 Unity Catalog에 의해 관리되길 원합니다.

이 특성 테이블을 만드는 올바른 방법은 무엇입니까?

A. 평소와 같이 데이터가 있는 Delta 테이블을 만든 다음 Python의 **`FeatureStoreClient`** 의 **`register_table`** 메서드를 사용하여 Unity Catalog에 feature table로 등록합니다.
B. SQL을 통해 **`AS FEATURE STORE`** 절을 사용하여 Unity Catalog에 빈 Delta 테이블을 만든 다음, 여기에 데이터를 씁니다.
C. Python에서 FeatureEngineeringClient의 **`create_table`** 메서드를 사용하여 테이블을 생성한 다음, 여기에 데이터를 씁니다.
D. Unity Catalog에 데이터가 있는 Delta 테이블을 만든 다음, SQL에서 **`ALTER TABLE`** 명령을 사용하여 **`SET AS FEATURE STORE`** 절을 사용해 이를 특성 테이블로 구성합니다.

질문 **2**

목표: 누락된 값을 최빈값, 평균 또는 중앙값으로 대체합니다.

데이터 과학자는 연속형 특성에서 누락된 값을 대체해야 합니다. 그들은 최소한의 노력으로 올바른 결과를 얻고 싶어합니다.

어떤 전략을 사용해야 합니까?

A. 특성 분포에 따라 최상의 방법론을 자동으로 선택하는 sklearn **`SimpleImputer`** 를 사용하세요.
B. 값의 분포를 검토하고 적절한 대체 값을 선택하십시오.
C. 연속형 열에 가장 적합한 대체값인 .mean()을 사용합니다.
D. 연속형 열에 가장 적합한 대체값인 .mode()를 사용합니다.

질문 **3**

목표: 훈련 데이터의 데이터 불균형을 완화하는 방법을 식별합니다.

데이터 과학자는 고객이 구독 서비스에서 이탈할지 여부를 예측하는 모델을 개발하는 머신 러닝 프로젝트를 진행하고 있습니다. 데이터 세트는 매우 불균형하며, 이탈하는 고객을 나타내는 인스턴스는 10%에 불과합니다. 그들은 귀하의 모델이 다수 클래스에 편향되지 않으면서도 소수 클래스을 효과적으로 식별하는지 확인하고 싶어합니다.

클래스 불균형으로 인해 이탈하지 않은 고객에 대한 모델의 편향을 직접적으로 완화하는 전략은 무엇입니까?

A. 특성을 정규화하여 동일한 규모에 있도록 하고, 이를 통해 모델 성능을 개선합니다.
B. 모델 학습 중에 소수 클래스에 더 높은 오분류 비용을 할당하여 비용 민감한 학습을 사용합니다.
C. 이탈하지 않은 고객에 대한 데이터를 더 많이 수집하여 학습 데이터 세트의 크기를 늘립니다.
D. 과적합을 줄이기 위해 더 간단한 모델을 사용하여 소수 클래스에 대한 일반화 성능을 보장합니다.

질문 **4**

목표: 그리드 검색 및 교차 검증 과정을 함께 수행할 때 학습되는 모델의 수를 식별합니다.

데이터 과학자는 5겹 교차 검증과 **`GridSearchCV`** 를 사용하여 scikit-learn에서 Support Vector Machine(SVM) 모델을 튜닝하고 있습니다. 파라미터 그리드에는 최적화를 위한 세 가지 하이퍼파라미터가 포함되어 있습니다. C는 값 **`[0.1, 1, 10]`**을, Kernel은 **`['linear', 'rbf']`**선택지를, gamma는 **`[0.01, 0.1, 1]`**값을 갖습니다.

총 몇 개의 다른 모델이 훈련될까요?

A. 90
B. 18
C. 1
D. 위의 어느 것도 아닙니다.

질문 **5**

목표: *Delta Live Tables*를 사용하여 스트리밍 추론이 어떻게 수행되는지 파악합니다.

어떤 회사는 수천 명의 사용자를 보유한 팟캐스트 플랫폼을 운영하고 있습니다. 이 회사는 사용자의 팟캐스트 청취, 일시 정지, 종료와 같은 이벤트를 10분 단위로 관찰하여 팟캐스트 참여도가 낮은 경우를 감지하는 이상 탐지 알고리즘을 구현했습니다. 머신 러닝 엔지니어는 초당 최대 수만 개의 이벤트를 처리해야 하는 프로덕션 데이터 파이프라인에 이 모델을 배포하려고 합니다. 이벤트 볼륨이 하루 종일 변동하므로 엔지니어는 파이프라인 컴퓨팅 크기를 동적으로 조정해야 합니다.

이러한 요구 사항을 충족하는 파이프라인 설계 방법은 무엇입니까?

A. 알고리즘을 Spark UDF로 적용하는 Delta Live Tables 파이프라인을 만듭니다.
B. 알고리즘을 Spark UDF로 적용하는 구조화된 스트리밍 작업을 생성합니다.
C. 모델 서빙 엔드포인트를 만들고, 해당 엔드포인트를 호출하는 사용자 지정 UDF를 실행하는 Delta Live Tables 파이프라인을 만듭니다.

D. 모델 서빙 엔드포인트를 만들고, 해당 엔드포인트를 호출하는 사용자 지정 UDF를 실행하는 구조화된 스트리밍 작업을 만듭니다.

답변
질문 1: C
질문 2: B
질문 3: B
질문 4: A
질문 5: A