Tutorial 2: Machine Learning

Databricks Free Edition

Feature Engineering, MLflow, Model Registry
e Batch Inference

Features >>> Train >>> Register >>> Predict

Autor: Thiago Charchar & Claude AI

Data: Fevereiro 2026

github.com/tlcharchar/databricks-medallion

Pre-requisito: Tutorial 1 (Medallion Architecture)

Sumario

1. Visao Geral

Este tutorial e a continuacao do Tutorial 1 (Medallion Architecture). Usamos a tabela Silver (taxi_trips_cleaned) como base para construir um pipeline de Machine Learning completo.

Caso de Uso

Prever o valor da gorjeta (tip_amount) de corridas de taxi em NYC.

Pergunta de negocio: "Dado uma corrida de taxi (distancia, hora, dia, zona), qual sera a gorjeta?"

Por que este caso de uso?

- E um problema de regressao (prever valor continuo)
- Usa dados que ja existem na Silver (sem nova ingestao)
- Tem features ricas (temporais, geograficas, de viagem)
- Resultado e intuitivo e verificavel

2. Arquitetura ML

```
Silver (taxi_trips_cleaned)
   |
   v
Feature Engineering (04)
   |  --> medallion_ml.taxi_tip_features
   |  --> medallion_ml.taxi_tip_train
   |  --> medallion_ml.taxi_tip_test
   v
AutoML Baseline (05)
   |  --> MLflow: 4 modelos (LR, DT, RF, GB)
   v
Hyperparameter Tuning (06)
   |  --> MLflow: nested runs, grid search
   |  --> medallion_ml.best_model_ref
   v
Model Registry via MLflow (07)
   |  --> runs:/{RUN_ID}/model (MLflow artifacts)
   |  --> medallion_ml.best_model_ref (atualizada)
   v
Batch Inference (08)
   |  --> medallion_gold.taxi_tip_predictions
   v
Genie Space
   --> "Qual a gorjeta prevista para corridas noturnas?"
```

---

3. Capacidades ML no Free Edition

O que esta disponivel

Recurso	Status	Como usamos
MLflow Experiments	Disponivel	Tracking de parametros e metricas
MLflow Model Logging	Disponivel	Salvar modelos como artefatos
MLflow Artifact Storage	Disponivel	Carregar modelos via runs:/ URI
Serverless Compute	Disponivel	Executar notebooks de ML
Genie Spaces	Disponivel	NLP sobre predicoes

O que NAO esta disponivel

Recurso	Status	Alternativa
Databricks AutoML	Nao disponivel	sklearn + MLflow manual
ML Runtime clusters	Nao disponivel	Serverless com pip install
Feature Store API	Legacy desabilitada	Delta Tables no Unity Catalog
Model Serving endpoints	Nao disponivel	Batch inference
UC Model Registry (upl..	Bloqueado (S3)	MLflow artifacts + Delta ref

Nota sobre Unity Catalog Model Registry

O Free Edition utiliza storage S3 gerenciado pela Databricks. Embora o Unity Catalog Models apareca como disponivel na interface, o upload de modelos para o registry falha com erro S3 AccessDenied (PutObject) porque as politicas de storage nao permitem escrita no bucket de artefatos. A alternativa e usar os artefatos do MLflow Tracking diretamente (runs:/{RUN_ID}/model), que e exatamente como muitas empresas operam com MLflow open-source.

---

4. Fase 1 -- Feature Engineering

Notebook: 04_feature_engineering.py

Transforma dados da Silver em features prontas para ML.

Categorias de Features

Temporais -- quando a corrida aconteceu:

```python
# Periodo do dia
.withColumn(
    "period_of_day",
    when((col("pickup_hour") >= 0) & (col("pickup_hour") < 6), 0)
    .when((col("pickup_hour") >= 6) & (col("pickup_hour") < 12), 1)
    .when((col("pickup_hour") >= 12) & (col("pickup_hour") < 18), 2)
    .otherwise(3)
)
# Fim de semana
.withColumn(
    "is_weekend",
    when(col("pickup_day_of_week").isin(1, 7), 1).otherwise(0)
)
# Horario de pico
.withColumn(
    "is_rush_hour",
    when(
        (col("is_weekend") == 0) &
        (col("pickup_hour").between(7, 9) |
         col("pickup_hour").between(17, 19)),
        1
    ).otherwise(0)
)
```

Geograficas -- de onde para onde (zonas arredondadas):

```python
.withColumn("pickup_zone_lat",
            spark_round(col("pickup_latitude"), 2))
.withColumn("pickup_zone_lon",
            spark_round(col("pickup_longitude"), 2))
```

De viagem -- caracteristicas da corrida:

```python
# Log-transform para normalizar distribuicoes
.withColumn("log_trip_distance", log1p(col("trip_distance")))
.withColumn("log_trip_duration", log1p(col("trip_duration_min")))
# Proxy de trafego
.withColumn("distance_duration_ratio",
            col("trip_distance") / col("trip_duration_min"))
```

Filtro importante: apenas cartao de credito

Gorjetas em dinheiro nao sao registradas no dataset. Filtramos apenas corridas pagas com cartao:

```python
.filter(col("payment_is_credit") == 1)
```

Train/Test Split

```python
df_train, df_test = df_features.randomSplit([0.8, 0.2], seed=42)
```

Tabelas criadas

Tabela	Descricao
medallion_ml.taxi_tip_fea..	Feature table completa
medallion_ml.taxi_tip_train	80% para treinamento
medallion_ml.taxi_tip_test	20% para avaliacao

---

5. Fase 2 -- AutoML Baseline

Notebook: 05_automl_baseline.py

Testa 4 algoritmos automaticamente e loga tudo no MLflow.

Modelos testados

Modelo	Tipo	Complexidade
LinearRegression	Linear	Baixa (baseline)
DecisionTree	Arvore	Media
RandomForest	Ensemble	Alta
GradientBoosting	Ensemble	Alta

Configuracao do MLflow

No serverless do Free Edition, e necessario configurar explicitamente o tracking URI e o registry:

```python
import mlflow

mlflow.set_registry_uri("databricks-uc")
os.environ["MLFLOW_TRACKING_URI"] = "databricks"

EXPERIMENT_NAME = "/Users/seu@email.com/nyc-taxi-tip-prediction"
mlflow.set_experiment(EXPERIMENT_NAME)
```

Loop de treinamento

```python
for model_name, model in models.items():
    with mlflow.start_run(run_name=f"automl_{model_name}"):
        mlflow.log_param("model_type", model_name)

        model.fit(X_train, y_train)

        y_pred = model.predict(X_test)
        mae = mean_absolute_error(y_test, y_pred)

        mlflow.log_metric("mae", mae)
        mlflow.log_metric("rmse", rmse)
        mlflow.log_metric("r2", r2)

        # Signature obrigatoria para Unity Catalog
        from mlflow.models import infer_signature
        signature = infer_signature(X_test, y_pred)
        mlflow.sklearn.log_model(
            model, "model",
            signature=signature,
            input_example=X_test[:5],
        )
```

Metricas de avaliacao

Metrica	Significado
MAE	Erro medio absoluto (em dolares)
RMSE	Raiz do erro quadratico medio
R2	Variancia explicada (0 a 1)

Nota sobre amostragem e memoria

O serverless do Free Edition tem limite de aproximadamente 2GB de RAM. Para evitar erros de memoria (SIGKILL exit code 137), amostramos os dados:

```python
MAX_TRAIN_ROWS = 50_000
MAX_TEST_ROWS = 10_000
if train_count > MAX_TRAIN_ROWS:
    sample_fraction = MAX_TRAIN_ROWS / train_count
    df_train_pd = df_train_spark.sample(
        fraction=sample_fraction, seed=42
    ).toPandas()
```

Tambem e importante usar n_jobs=1 no RandomForest para evitar forks de memoria:

```python
RandomForestRegressor(
    n_estimators=50, max_depth=8,
    n_jobs=1, random_state=42
)
```

Nota sobre Model Signature

O Unity Catalog exige que modelos tenham signature (schema de input/output). Sem ela, o registro falha com erro. Use infer_signature:

```python
from mlflow.models import infer_signature
signature = infer_signature(X_test, y_pred)
mlflow.sklearn.log_model(model, "model", signature=signature)
```

---

6. Fase 3 -- Hyperparameter Tuning com MLflow

Notebook: 06_mlflow_training.py

Otimiza o melhor modelo (GradientBoosting) com grid search.

Grid de hiperparametros

```python
param_grid = {
    "n_estimators": [50, 100],
    "max_depth": [4, 6],
    "learning_rate": [0.05, 0.1],
    "subsample": [0.8],
}
# Total: 8 combinacoes
```

O grid foi reduzido para 8 combinacoes (vs 12 original) para respeitar os limites de memoria do serverless Free Edition.

Nested Runs no MLflow

Usamos um parent run que organiza todos os child runs:

```python
with mlflow.start_run(run_name="hyperparameter_tuning_gb") as parent:
    for params in grid:
        with mlflow.start_run(nested=True) as child:
            model = GradientBoostingRegressor(**params, random_state=42)
            model.fit(X_train, y_train)

            # Metricas
            mlflow.log_metric("mae", mae)

            # Modelo com signature
            signature = infer_signature(X_test, y_pred)
            mlflow.sklearn.log_model(model, "model", signature=signature)
```

Salvar referencia do melhor modelo

O notebook salva o run_id do melhor modelo numa tabela Delta para uso nos proximos notebooks:

```python
spark.sql(f"""
    CREATE OR REPLACE TABLE {CATALOG}.{SCHEMA_ML}.best_model_ref AS
    SELECT
        '{best_run_id}' as run_id,
        '{parent_run_id}' as parent_run_id,
        '{EXPERIMENT_NAME}' as experiment_name,
        {best_mae} as best_mae,
        current_timestamp() as registered_at
""")
```

Visualizacao no MLflow UI

No menu lateral do Databricks, va em Experiments e abra nyc-taxi-tip-prediction. Voce vera:

- Parent run com 8 child runs
- Graficos comparativos de metricas
- Tabela com todos os hiperparametros

---

7. Fase 4 -- Model Registry via MLflow Tracking

Notebook: 07_model_registry.py

Seleciona o melhor modelo e prepara para inferencia.

Por que nao usar o UC Model Registry?

No Free Edition, o upload de modelos para o Unity Catalog Model Registry falha com erro de S3 AccessDenied. A solucao e usar o MLflow Tracking diretamente para gerenciar modelos -- carregando-os via URI de artefatos (runs:/{RUN_ID}/model).

Em edicoes pagas do Databricks, voce usaria:

```python
mlflow.set_registry_uri("databricks-uc")
mlflow.register_model(model_uri, "catalog.schema.model_name")
client.set_registered_model_alias(name, "Champion", version)
```

Fluxo do notebook

```
best_model_ref (Delta table)
    |
    v
Recuperar run_id do melhor modelo
    |
    v
Consultar detalhes no MLflow (MlflowClient)
    |
    v
Listar e comparar todos os runs
    |
    v
Carregar modelo via runs:/{RUN_ID}/model
    |
    v
Testar predicoes com amostra
    |
    v
Atualizar best_model_ref com model_uri
```

Consultar runs do MLflow

```python
from mlflow.tracking import MlflowClient

client = MlflowClient()
experiment = client.get_experiment_by_name(EXPERIMENT_NAME)
runs = client.search_runs(
    experiment_ids=[experiment.experiment_id],
    order_by=["metrics.mae ASC"],
    max_results=20
)
```

Carregar modelo do MLflow

```python
model_uri = f"runs:/{best_run_id}/model"
model = mlflow.sklearn.load_model(model_uri)
```

Testar com dados reais

```python
df_test = spark.table(f"{CATALOG}.{SCHEMA_ML}.taxi_tip_test") \
    .limit(100).toPandas()

y_pred = model.predict(X_sample)
mae = mean_absolute_error(y_actual, y_pred)
within_1 = np.mean(np.abs(y_pred - y_actual) < 1.0) * 100
```

Salvar referencia para batch inference

O notebook atualiza a tabela best_model_ref adicionando o campo model_uri, que sera usado pelo notebook 08:

```python
spark.sql(f"""
    CREATE OR REPLACE TABLE {CATALOG}.{SCHEMA_ML}.best_model_ref AS
    SELECT
        '{best_run_id}' as run_id,
        '{model_uri}' as model_uri,
        '{EXPERIMENT_NAME}' as experiment_name,
        '{type(model).__name__}' as model_type,
        {best_mae} as best_mae,
        current_timestamp() as registered_at
""")
```

---

8. Fase 5 -- Batch Inference

Notebook: 08_batch_inference.py

Aplica o melhor modelo em batch para gerar predicoes.

Fluxo

```
Carregar model_uri de best_model_ref
    |
    v
mlflow.sklearn.load_model(model_uri)
    |
    v
Amostrar dados de teste (50K rows)
    |
    v
Gerar predicoes em Pandas
    |
    v
Calcular erros (predicted - actual)
    |
    v
Agregar em tabela Gold por hora/dia/periodo
    |
    v
taxi_tip_predictions (Gold)
```

Carregar modelo

python

```python
ref = spark.table(f"{CATALOG}.{SCHEMA_ML}.best_model_ref").collect()[0]
model_uri = ref["model_uri"]
model = mlflow.sklearn.load_model(model_uri)
```

Gerar predicoes

python

```python
MAX_INFERENCE_ROWS = 50_000
df_test = df_test_full.sample(fraction=sample_frac, seed=42)
df_test_pd = df_test.toPandas()

df_test_pd["predicted_tip"] = model.predict(df_test_pd[FEATURE_COLS].values)
df_test_pd["prediction_error"] = df_test_pd["predicted_tip"] - df_test_pd["target_tip_amount"]
df_test_pd["absolute_error"] = abs(df_test_pd["prediction_error"])
```

Analise de erro

O notebook calcula:

- MAE geral do modelo
- % de predicoes dentro de $1 e $2 de erro
- Erro por hora do dia (quando o modelo acerta mais?)
- Erro por periodo (madrugada, manha, tarde, noite)

Tabela Gold ML

```python
GOLD_ML_TABLE = "workspace.medallion_gold.taxi_tip_predictions"
```

Colunas principais:

Coluna	Descricao
avg_actual_tip	Gorjeta real media
avg_predicted_tip	Gorjeta prevista pelo modelo
avg_prediction_error	Erro medio do modelo
pct_accurate_within_1usd	% de predicoes com erro menor que $1
period_name	Periodo do dia (Madrugada, Manha, Tarde, Noite)
day_name	Nome do dia da semana

---

9. Fase 6 -- Genie Space com predicoes ML

Atualizar o Genie Space

No Genie Space criado no Tutorial 1, adicione a nova tabela:

1. Va em Genie no menu lateral
2. Edite o Space NYC Taxi Analytics
3. Adicione a tabela: workspace.medallion_gold.taxi_tip_predictions
4. Atualize as General instructions adicionando:

```
A tabela taxi_tip_predictions contem predicoes de ML
sobre gorjetas feitas por um modelo GradientBoosting.

Colunas de ML:
- avg_actual_tip = gorjeta real media
- avg_predicted_tip = gorjeta prevista pelo modelo
- avg_prediction_error = erro medio da previsao
- pct_accurate_within_1usd = % de previsoes precisas
- period_name = Madrugada/Manha/Tarde/Noite

O modelo foi treinado com features temporais, geograficas
e de viagem. Use estas colunas para responder perguntas
sobre predicoes e acuracia do modelo.
```

Perguntas para testar

- "Qual a gorjeta prevista para corridas noturnas?"
- "Em que horario o modelo erra mais?"
- "Qual o percentual de acerto do modelo nos fins de semana?"
- "Compare a gorjeta real vs prevista por periodo do dia"

- "Quando as gorjetas sao maiores: manha ou noite?"

---

10. Troubleshooting

Erro: SIGKILL exit code 137 (Out of Memory)

Causa: Dataset muito grande para a memoria do serverless (~2GB RAM). Solucao: Reduzir o numero de amostras e a complexidade dos modelos:

```python
MAX_TRAIN_ROWS = 50_000  # nao usar 500K+
MAX_TEST_ROWS = 10_000
sample_frac = min(1.0, MAX_ROWS / total_count)
df_pd = df_spark.sample(fraction=sample_frac, seed=42).toPandas()
```

Tambem ajuda reduzir n_estimators, max_depth, e usar n_jobs=1.

Erro: S3 AccessDenied (PutObject) ao registrar modelo no UC

Causa: O Free Edition nao permite upload de artefatos para o bucket S3 do Unity Catalog Model Registry. Solucao: Usar artefatos do MLflow Tracking diretamente:

```python
# Em vez de:
mlflow.register_model(model_uri, "catalog.schema.model_name")

# Usar:
model_uri = f"runs:/{best_run_id}/model"
model = mlflow.sklearn.load_model(model_uri)
```

Salve o model_uri numa tabela Delta para referencia entre notebooks.

Erro: Model did not contain any signature metadata

Causa: Unity Catalog exige que modelos tenham signature. Solucao: Adicionar infer_signature ao logar o modelo:

```python
from mlflow.models import infer_signature
signature = infer_signature(X_test, y_pred)
mlflow.sklearn.log_model(model, "model", signature=signature)
```

Erro: CONFIG_NOT_AVAILABLE (spark.mlflow.modelRegistryUri)

Causa: No serverless, a config do MLflow precisa ser setada via Python, nao via Spark config. Solucao: Adicionar antes de set_experiment:

```python
mlflow.set_registry_uri("databricks-uc")
os.environ["MLFLOW_TRACKING_URI"] = "databricks"
```

Erro: legacy workspace model registry is disabled

Causa: Free Edition usa Unity Catalog, nao o registry antigo. Solucao: Configurar MLflow para usar UC:

```python
mlflow.set_registry_uri("databricks-uc")
MODEL_NAME = "catalog.schema.model_name"  # formato 3 niveis
```

Erro: Only serverless compute is supported

Causa: Free Edition nao permite clusters customizados. Solucao: Usar sklearn (single-node) em vez de SparkML (distribuido). O serverless suporta bem com datasets amostrados.

---

11. Proximos Passos

Apos completar este tutorial, considere:

1. SparkML -- Substituir sklearn por SparkML para treinar de forma distribuida
2. Feature Store -- Quando disponivel, migrar features para o Feature Store nativo
3. Model Serving -- Em edicoes pagas, criar endpoints REST para inferencia online
4. UC Model Registry -- Em edicoes pagas, registrar modelos com aliases Champion/Challenger
5. Retraining -- Automatizar retraining periodico com Workflows
6. Monitoring -- Implementar data drift e model drift detection
7. Deep Learning -- Explorar PyTorch/TensorFlow para modelos mais complexos
8. LLMs -- Usar os modelos do Playground/Mosaic AI para tarefas de NLP

Resumo: Estrutura completa do Projeto

```
databricks-medallion/
|-- notebooks/
|   |-- 01_bronze_ingestion.py         # Tutorial 1
|   |-- 02_silver_transformation.py    # Tutorial 1
|   |-- 03_gold_aggregation.py         # Tutorial 1
|   |-- 04_feature_engineering.py      # Tutorial 2
|   |-- 05_automl_baseline.py          # Tutorial 2
|   |-- 06_mlflow_training.py          # Tutorial 2
|   |-- 07_model_registry.py           # Tutorial 2
|   |-- 08_batch_inference.py          # Tutorial 2
|-- sql/
|   |-- gold_queries.sql
|-- config/
|   |-- pipeline_config.yaml
|-- README.md
```

Tabelas criadas (Tutorial 2)

Camada	Tabela	Descricao
ML	medallion_ml.taxi_tip_..	Feature table completa
ML	medallion_ml.taxi_tip_..	Dados de treinamento (80%)
ML	medallion_ml.taxi_tip_..	Dados de teste (20%)
ML	medallion_ml.best_mode..	Referencia ao melhor modelo (run_i..
Gold	medallion_gold.taxi_ti..	Predicoes agregadas

Como os modelos sao gerenciados

```
MLflow Experiment (nyc-taxi-tip-prediction)
   |
   |-- automl_LinearRegression
   |-- automl_DecisionTree
   |-- automl_RandomForest
   |-- automl_GradientBoosting
   |-- hyperparameter_tuning_gb (parent)
   |   |-- gb_lr0.05_d4_n50
   |   |-- gb_lr0.05_d4_n100
   |   |-- gb_lr0.05_d6_n50
   |   |-- ... (8 child runs total)
   |
   v
best_model_ref (Delta table)
   |-- run_id: referencia ao melhor run
   |-- model_uri: runs:/{RUN_ID}/model
   |-- best_mae: metrica do melhor modelo
```

Tutorial criado como parte do projeto databricks-medallion. Repositorio: github.com/tlcharchar/databricks-medallion