

New

Workspace

Recents

Catalog

Workflows

Compute

SQL

SQL Editor

Queries

Dashboards

Genie

Alerts

Query History

SQL Warehouses

Data Engineering

Job Runs

Data Ingestion

Delta Live Tables

Machine Learning

Playground

Experiments

Untitled Notebook 2024-12-07 23:24:50 Python

File Edit View Run Help

Last edit was 7 minutes ago

Run all

SANTHANA KRISHNAN

Schedule

Share

11:28 PM (8s)

Python

```
# Databricks notebook containing ETL, ML training, and deployment steps.

# Step 1: Initialize Spark Session
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import joblib
import numpy as np
import pandas as pd
from azure.storage.blob import BlobServiceClient
import io
import os

spark = SparkSession.builder \
    .appName("ETL ML Training and Deployment") \
    .getOrCreate()

# Step 2: ETL Process (Extract, Transform, Load)

# Example: Extracting data from an external CSV file or direct data input
new_data = [
    (1.5, 2.3, 3.1, 4.0, 5.2),
    (1.4, 2.2, 3.0, 4.1, 5.1),
    (1.7, 2.5, 3.3, 4.3, 5.4),
    (1.8, 2.6, 3.4, 4.4, 5.5),
]

# Define the column names for the data
columns = ['feature1', 'feature2', 'feature3', 'feature4', 'feature5']

# Step 3: Load data into a PySpark DataFrame
new_data_df = spark.createDataFrame(new_data, columns)

# Step 4: Feature Engineering (Transform Data)

# Using VectorAssembler to combine features into a single vector column
assembler = VectorAssembler(inputCols=columns, outputCol="features")
assembled_data = assembler.transform(new_data_df)

# Step 5: Machine Learning - Train a Model (Linear Regression)

# For simplicity, we will use random data for training as an example
X = np.random.rand(100, 5) # Random features
y = np.random.rand(100) # Random target values

# Split the data for training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse}")

# Step 6: Save the trained model to Azure Blob Storage

# Azure Blob Storage details
connection_string = "DefaultEndpointsProtocol=https;AccountName=ijazblob1;
AccountKey=0Hedg6WdUSruycWuXnyjZceDUXpNk3Mm8fBEFzQFq7mOzGyeH3jV5Q2tKfWunaEaXhQkdT+7jYt+AStrnKF6Q==;EndpointSuffix=core.windows.net"
container_name = "ijaz-container"
model_blob_name = "linear_regression_model.pkl" # Model filename in the container

# Initialize BlobServiceClient
blob_service_client = BlobServiceClient.from_connection_string(connection_string)

# Get BlobClient to upload the model file
blob_client = blob_service_client.get_blob_client(container=container_name, blob=model_blob_name)

# Ensure the directory exists
os.makedirs("/dbfs/tmp", exist_ok=True)

# Save the model as a .pkl file
with open("/dbfs/tmp/linear_regression_model.pkl", "wb") as model_file:
    joblib.dump(model, model_file)

# Upload the saved model file to Azure Blob Storage
with open("/dbfs/tmp/linear_regression_model.pkl", "rb") as model_file:
    blob_client.upload_blob(model_file, overwrite=True)

print(f"Model uploaded successfully to Azure Blob Storage as {model_blob_name}")

# Step 7: Predicting with the trained model on new data

# Generate predictions (in a real-world scenario, this would use new data)
def generate_predictions(model, assembled_data):
    # Convert assembled data to pandas and get features
```

```

pandas_df = assembled_data.select('features').toPandas()
features = np.array(pandas_df['features'].tolist())

# Use the trained model to predict
predictions = model.predict(features)

# Convert predictions back to a Spark DataFrame
prediction_df = spark.createDataFrame(pd.DataFrame({'prediction': predictions}))
return prediction_df

# Generate predictions using the trained model
prediction_df = generate_predictions(model, assembled_data)

# Show predictions
display(prediction_df)

# Step 8: Save the predictions to Azure Blob Storage as a CSV file

# Convert the Spark DataFrame to Pandas to write as CSV
prediction_pandas_df = prediction_df.toPandas()

# Save the predictions to a CSV file in Azure Blob Storage
predictions_blob_name = "predictions.csv"
predictions_blob_client = blob_service_client.get_blob_client(container=container_name, blob=predictions_blob_name)

# Write the CSV to Blob Storage
with io.BytesIO() as output:
    prediction_pandas_df.to_csv(output, index=False)
    output.seek(0)
    predictions_blob_client.upload_blob(output, overwrite=True)





print(f"Predictions saved to Azure Blob Storage as {predictions_blob_name}")

```

▶ (2) Spark Jobs

▼ (1) MLflow run

Logged 1 run [🔗](#) to an experiment [🔗](#) in MLflow. [Learn more](#) [🔗](#)

- ▶  assembled_data: pyspark.sql.dataframe.DataFrame
- ▶  new_data_df: pyspark.sql.dataframe.DataFrame = [feature1: double, feature2: double ... 3 more fields]
- ▶  prediction_df: pyspark.sql.dataframe.DataFrame = [prediction: double]
- ▶  prediction_pandas_df: pandas.core.frame.DataFrame = [prediction: float64]

Uploading artifacts: 100% 9/9 [00:00<00:00, 2.75it/s]

Mean Squared Error (MSE): 0.0758269903258609

Model uploaded successfully to Azure Blob Storage as linear_regression_model.pkl

Table ▼ +

[🔍](#) [🔧](#) [📄](#)

	1.2 prediction
1	0.58732500635712...
2	0.58485734088814...
3	0.54643862443594...
4	0.530577604761266

⬇ 4 rows | 7.52 seconds runtime

Refreshed 7 minutes ago

Predictions saved to Azure Blob Storage as predictions.csv

[Shift+Enter] to run and move to next cell
 [Ctrl+Shift+P] to open the command palette
 [Esc H] to see all keyboard shortcuts