Microsoft

Designing and Implementing a Data Science Solution on Azure

Exam Ref DP-100

Dayne Sorvisto

Microsoft

Exam Ref DP-100 Designing and Implementing a Data Science Solution on Azure

Dayne Sorvisto

CHAPTER 2

Explore data and train models

We're about to dive deep into the heart of the machine learning lifecycle: exploring data and training models. This chapter is designed to equip you with the skills and knowledge needed to handle data effectively and create optimized machine learning models using Azure Machine Learning.

Data exploration is the first step in any successful machine learning project. It's where you'll get to know your data, understand its characteristics, and prepare it for modeling. You'll learn how to access and wrangle data using Azure's data assets and datastores, making your data ready for the challenges ahead.

Model training is where the magic happens. You'll discover how to create models using the Azure Machine Learning Designer, leverage the power of automated machine learning for various data types, and even dive into custom model training using notebooks and Python SDKv2. We'll also cover hyperparameter tuning, a crucial step in optimizing your models for better performance.

By the end of this chapter, you'll have a solid understanding of how to explore data and train models in Azure Machine Learning, setting the stage for deploying and managing your models in the real world.

EXAM TIP

Follow the steps outlined in Skill 2.1 and Skill 2.2 to understand how to create models, data assets, and datastores in Azure Machine Learning Designer and by using the Python SDKv2. While you may prefer one over the other, pay close attention to the wording on the exam when a question asks about the Azure Machine Learning Designer or the SDK since it might impact how you answer the question.

Skills covered in this chapter:

- Skill 2.1: Explore data by using data assets and datastores
- Skill 2.2: Create models by using the Azure Machine Learning Designer
- Skill 2.3: Use automated machine learning to explore optimal models
- Skill 2.4: Use notebooks for custom model training
- Skill 2.5: Tune hyperparameters with Azure Machine Learning

As you journey through this chapter, remember that exploring data and training models are iterative processes. With each iteration, you'll gain deeper insights into your data and refine your models for better accuracy and performance. After acquiring the five skills in this chapter, you will be able to combine them to build training pipelines using automated machine learning and tune hyperparameters to iteratively improve the model performance using Azure Machine Learning.

Skill 2.1: Explore data by using data assets and datastores

In the process of developing a machine learning model, one of the first steps is exploring and understanding the data you're working with. Skill 2.1 focuses on the exploration of data using Azure Machine Learning's data assets and datastores. This skill is essential for data scientists and analysts who need to access, wrangle, and prepare data for model training. By mastering these techniques, you'll be able to create a solid foundation for building accurate and efficient machine learning models.

This skill covers how to:

- Access and wrangle data during interactive development
- Wrangle interactive data with Apache Spark

Access and wrangle data during interactive development

In this section, you'll learn how to access data stored in Azure Machine Learning datastores and perform data wrangling operations interactively. This is crucial for exploratory data analysis and preprocessing steps before model training.

Imagine you're working on a project to predict customer churn based on historical transaction data. You need to access this data from an Azure Blob Storage, clean it, and perform feature engineering to prepare it for model training. You'll use Python in a Jupyter Notebook environment within Azure Machine Learning to load the data, handle missing values, encode categorical variables, and normalize numerical features.

NEED MORE REVIEW? **WRANGLING DATA IN AZURE MACHINE LEARNING**

You can read more about wrangling data in Azure Machine learning at *https://learn.microsoft.com/en-us/azure/machine-learning/how-to-access-data-interactives*

To demonstrate the end-to-end process of exploring data and training a model to predict customer churn using Azure Machine Learning, we'll go through the following steps:

1. Before you begin, make sure you have an Azure Machine Learning workspace set up. You can create one using the Azure portal or the Azure Machine Learning SDK. For detailed instructions, see the section "Manage an Azure Machine Learning workspace" in Chapter 1.
2. Create a datastore: Link your Azure Blob Storage account to your Azure Machine Learning workspace by creating a datastore. For a review of managing data in Azure Machine Learning, see Chapter 1, "Manage data in an Azure Machine Learning workspace." For convenience, here are the instructions for creating a datastore:
    1. Navigate to the Datastores section in the left menu and select the Datastores option under the Data section.
    2. Add a new datastore by clicking the New Datastore (+) button at the top of the Datastores page.
3. Access and explore data: Use the Azure Machine Learning SDK to access your data and perform exploratory data analysis (EDA) using Pandas.
4. Preprocess and prepare data: Clean the data, handle missing values, encode categorical variables, and normalize numerical features.

Listing 2-1 shows a sample code snippet that demonstrates these steps: Setting up your Azure Machine Learning workspace, creating and accessing a datastore, and preprocessing data. Next, we will show how to implement these steps using `Workspace` and `Datastore` objects.

LISTING 2-1 Implementing preprocessing steps needed to train a model

```
from azureml.core import Workspace, Datastore
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Set up your Azure Machine Learning workspace
subscription_id = 'your-subscription-id'
resource_group = 'your-resource-group'
workspace_name = 'your-workspace-name'
workspace = Workspace(subscription_id, resource_group, workspace_name)

# Create a datastore (if not already created)
datastore_name = 'your-datastore-name'
container_name = 'your-container-name'
account_name = 'your-storage-account-name'
datastore = Datastore.register_azure_blob_container(workspace=workspace,
datastore_name=datastore_name,
container_name=container_name,
account_name=account_name)
```

```
# Access data from the datastore
datastore_path = [(datastore, 'path/to/your/data.csv')]
data = Dataset.Tabular.from_delimited_files(path=datastore_path)
df = data.to_pandas_dataframe()

# Explore and preprocess the data (assuming 'Churn' is the target variable and it's a
binary classification problem)
df.fillna(df.mean(), inplace=True)  # Handle missing values
df = pd.get_dummies(df, drop_first=True)  # Encode categorical variables
X = df.drop('Churn', axis=1)
y = df['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Normalize the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train a Logistic Regression model we imported
logreg = LogisticRegression()
logreg.fit(X_train_scaled, y_train)
# Predict on the test set
y_pred = logreg.predict(X_test_scaled)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred) print(f'Accuracy: {accuracy:.2f}')
```

In this example, we've used a logistic regression model for simplicity, but you can replace it with any other model suitable for your use case. Remember to adjust the data preprocessing steps according to the specific requirements of your dataset and the model you choose.

Wrangle interactive data with Apache Spark

Apache Spark is a powerful tool for handling large-scale data processing and analysis. In this topic, you'll explore how to use Apache Spark within Azure Machine Learning to wrangle data interactively.

Consider a scenario where you're dealing with a massive dataset of social media posts, and you need to perform sentiment analysis. The dataset is too large to process on a single machine, so you decide to use Apache Spark to process and clean the data in parallel. The decision to use Spark is reasonable if you have more than 20 GB of data, for example, where the data is too large to fit completely in memory on a single machine. You can also use Spark for much larger data volumes up to petabytes of data. You'll learn how to initialize a Spark session in Azure Machine Learning, read the data, and perform text preprocessing tasks like tokenization, stopword removal, and stemming. Figure 2-1 shows the workflow.

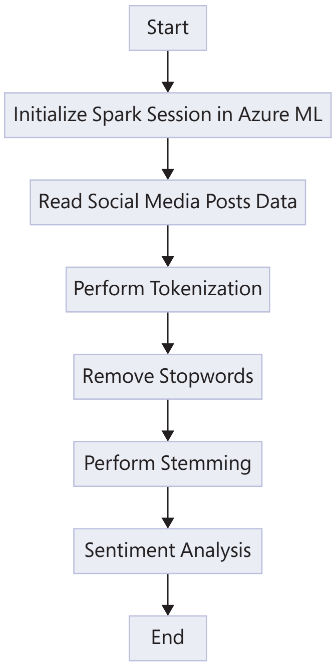

FIGURE 2-1 Workflow for sentiment analysis solution in Apache Spark

To use Apache Spark pools within Azure Machine Learning for wrangling a large dataset of social media posts stored in a datastore, you can follow these steps, which include creating a Spark pool and creating a datastore.

1. Set Up Your Azure Machine Learning Workspace: Make sure you have an Azure Machine Learning workspace set up.
2. Create a Datastore: Link your Azure Blob Storage account (where your social media posts dataset is stored) to your Azure Machine Learning workspace by creating a datastore.
3. Create a Spark Pool: In the Azure portal, navigate to your Azure Synapse Analytics workspace and create a Spark pool. You can check to make sure the Spark pool is correctly configured by verifying the node count, node size, and other settings on the Spark pool configuration page.

If you followed the above instructions to configure your datastore and have created a Spark pool, then you are ready to use the Spark pool to read the large dataset from the datastore, perform text preprocessing tasks like tokenization, stopword removal, and stemming, and prepare the data for sentiment analysis. Before reading the code in Listing 2-2, you should have a conceptual understanding of Spark's execution model to understand how model training can be distributed using a feature like Spark pools. Figure 2-2 illustrates how Spark's execution model is built on the concept of a directed acyclic graph (DAG) internally.

Listing 2-2 demonstrates how to use the Spark pool to read the large dataset from the datastore, perform text preprocessing tasks like tokenization, stopword removal, and stemming, and prepare the data for sentiment analysis.

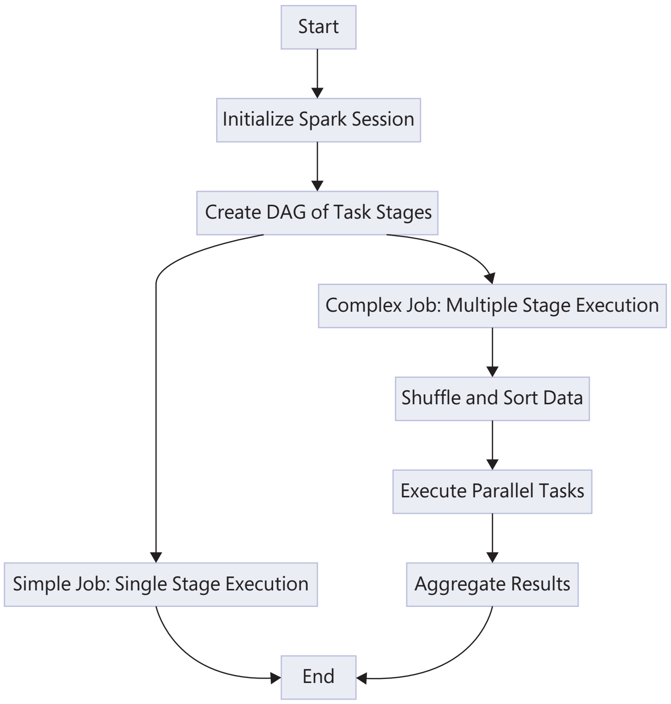
Start
Initialize Spark Session
Create DAG of Task Stages
Complex Job: Multiple Stage Execution
Shuffle and Sort Data
Execute Parallel Tasks
Simple Job: Single Stage Execution
Aggregate Results
End

FIGURE 2-2 Workflow using Spark's directed acyclic graph (DAG) for task execution

LISTING 2-2 Spark pool for reading large datasets from the datastore and preprocessing tasks

```
from azureml.core import Workspace, Datastore
from azureml.core.compute import SynapseCompute
from azureml.core.compute_target import ComputeTargetException
from pyspark.sql import SparkSession
from pyspark.ml.feature import Tokenizer, StopWordsRemover, HashingTF, IDF

# Set up your Azure Machine Learning workspace
subscription_id = 'your-subscription-id'
resource_group = 'your-resource-group'
workspace_name = 'your-workspace-name'
workspace = Workspace(subscription_id, resource_group, workspace_name)

# Create a Spark pool (if not already created)
spark_pool_name = "synapse-spark-pool"
try:
    spark_pool = SynapseCompute(workspace=workspace,
    name=spark_pool_name)
    print('Found existing Spark pool.')
except ComputeTargetException:
    print('Creating a new Spark pool.')
    spark_pool_config = SynapseCompute.
provisioning_configuration(compute_pool_name=spark_pool_name)
    spark_pool = ComputeTarget.create(workspace, spark_pool_name, spark_pool_config)
    spark_pool.wait_for_completion(show_output=True)
```