The **init()** and **run()** functions are required for the entry script to work. The **start()** function and the **initialize()** function are not required for the entry script to work.

The code segment that includes **"version1=90, version2=10"** sends 10 percent of incoming requests to version2 and 90 percent to version1. The code segment that includes **"version1=10, version=90"** sends 90 percent of incoming requests to version2. The code segment that includes **--mirror-traffic** sends 100 percent of incoming requests to more than one endpoint. The code segment that includes **--name** renames the endpoint.

The traffic parameter is used to configure traffic between two or more deployments but cannot be used to mirror traffic between them. The kind parameter is used to define the kind of endpoint, such as K8 or managed online endpoint. The tags parameter is used to list any tags associated with the endpoint. Only the **mirror_traffic** parameter can be used to duplicate a percentage of traffic to both deployments.

Running the **mlflow.autolog()** command will enable automatic logging. Using the print command will log the argument of the method but will not enable automatic logging. Using MLflow model registry is a centralized model store. but does not enable automatic logging. Running the **mlflow.set_tracking_uri()** command will set the MLflow tracking URI but not enable automatic logging.

The **Local feature importance** method returns feature importance for a single example. The **Precision method** is a measure of model performance. The **Global feature importance** method returns feature importance for the entire dataset. The **Gini coefficient method** is a criterion used in Decision Tree Classifiers.

To save each different version of trained models, the **register_model method** must be used. The **get_run method** returns a collection of metadata about the run. **log_artifact** and **log_param** are used to log elements of the run, but do not create a new version of the model in the model registry.

To attach to an already existing cluster by using Azure ML Python SDK v2, the **name of the cluster** must be used and then the appropriate method to connect to the cluster by passing the name of the cluster to it.

One of the points of a pipeline is to reduce cost by being able to use the proper machine for the proper component. **Memory heavy data processing benefits from a high memory CPU machine, while reducing the cost of the machine**. Simply **downgrading the GPU machines** would negatively impact performance. Finally, the other options don’t satisfy the requirement of GPU machine for model training.

A submitted job will appear in the submission list, along with its status. However, the pipeline job status and results are not filled back to the authoring page. Finally, the authoring page does not include details about pipeline job status.

In Azure ML Python SDK v2, to import the pipeline class, the below mentioned syntax has to be used. **“from azure.ai.ml.dsl import pipeline”.**

To pass data between two different pipeline steps, **OutputFileDatasetConfig class** is used.

To execute the script of a **custom component, metadata of the component, inputs, outputs, and compute environment** need to be mentioned in the **YAML file** of the custom component.

MLflow is a tool for managing the lifecycle of an ML model. **ML Pipelines** are executable workflows commonly used to train and deploy ML models. **Compute instances** are compute resources only, similar to a local laptop, but in the cloud. The **Responsible AI Dashboard** consists of many features, including the ability to conduct counterfactual what if analysis on input features.

The **Machine Learning designer** is a low/no code solution for developing machine learning pipelines. A **batch endpoint** is used to fulfill batch inference requests. **An Azure Blob Datastore** is a storage resource for storing data. **A Responsible AI scorecard** automatically generates cohort analysis reports, including MAE per cohort in the dataset.

Azure Machine Learning compute cluster is compatible with Azure Machine Learning designer. **Azure Databricks, a remote VM, and a local computer** are **not compatible** with the Azure Machine Learning designer.

When you have low-to-moderate data, no need of on-demand autoscaling, and also you have an Azure VM available with all the packages required for machine learning model training, selecting **attached compute and local compute** type are the most appropriate options.

You need to select an approach of training the best machine learning model which requires no coding and minimum skills in machine learning and that can only be fulfilled by **Azure AutoML**.

The **managed online endpoint** provides scalable real-time inference and doesn’t need infrastructure management. The **Kubernetes online endpoint** is also a scalable real-time option, but it requires your team to manage the infrastructure. The **batch endpoint** doesn’t provide real-time inference. And the **local web service** is not scalable, while still requiring your team to manage the infrastructure.

The integration process starts by **opening the terminal window** in the **AML Notebook tab**. Thereafter, a new SSH key must be generated, clone the public key to Git account, and clone the Git repository with the Git clone command.

The **min_instances** parameter must be set to one, and the **idle_time_before_scale_down** parameter must be set to 3600 for the instance count to scale down to one after an hour of inactivity.

To **create a workspace using Azure ML Python SDK v2**, the SDK must be installed first.

To connect to an already existing workspace using the configuration file, we have to use **“from_config()”** method from MLClient class of the Azure ML Python SDK.

To create a new Azure Machine Learning workspace and use an existing custom data encryption key, appropriate parameters need to be passed to the Workspace class

```
from azure.ai.ml.entities import Workspace, CustomerManagedKey ws = Workspace(
name="my_workspace", location="eastus", display_name="My workspace",
description="This example shows how to create a workspace",
customer_managed_key=CustomerManagedKey(
key_vault="/subscriptions/<SUBSCRIPTION_ID>/resourcegroups/<RESOURCE_GROUP>/provid
ers/microsoft.keyvault/vaults/<VAULT_NAME>" key_uri="<KEY-IDENTIFIER>" )
tags=dict(purpose="DP-100 Exam") ) ml_client.workspaces.begin_create(ws)
```

Mltable is the SDK v2 file type dedicated to tabular data. **FileDataset and TabularDataset** belong to the previous version of the SDK. While **uri_folder** is in the SDK v2, it is not the preferred data asset type for tabular data.

Curated environments are made available by Microsoft for use in Azure Machine Learning. They point to a specific cached **docker image**. The other options are either custom environments or curated ones that do not support PyTorch. `environment="AzureML-pytorch-1.8-ubuntu18.04-py37-cuda11-gpu@latest"`

The **Environment and Model classes** are not used to provision compute resources. The **ComputeInstance class** is used to provision a compute resource, but the resource can contain only one node. The **AmlCompute class** is used to provision a compute resource that can contain many nodes.

Refactoring the Jupyter notebook into .py file enables automated tests in the CI process. Writing a series of checks and publishing a Python pipeline are not part of the automated CI process. Refactoring the Jupyter notebooks into R scripts will not help in enabling automated checks in the CI process.

The **az ml model register command** is for saving a model in the AML workspace. The **az ml computetarget create command** is used to create a new compute target. The **az ml run submit-script command** is used to submit a script for execution. The **az ml model deploy command** is for deploying a model.

The **run method** of a batch scoring script is used for running inference with a loaded model. The **main method** is used to run a python file as a script. The **azureml_main method** is used to run a custom Python script in Azure Machine Learning designer. The **init method** of a batch scoring script is used for loading the model.

To deploy an online endpoint with the name of the endpoint, its instance type, its environment and its code configuration, among other things, the class **ManagedOnlineDeployment needs** to be used. The **class OnlineEndpoint** retrieves the online endpoint's property once deployed but does not allow it to deploy one.

The **BatchDeployment class** is used to create the deployment definition for a batch deployment. Similarly, for an **online deployment** you would use the OnlineDeployment. **Parallel and Pipeline classes** are used for creating components.

The **input parameter** is used to configure the path on a datastore. The **input-type parameter** is used to specify whether the input is a file or a folder. The **outputs parameter** is used to specify where to store the results. The **job-name parameter** is used to specify the name of the job for batch invoke.

The **Multiclass Decision Forest algorithm** will make a prediction between several categories. The **Bayesian Linear Regression** algorithm will forecast a numeric value, not make a prediction between several categories. The **Boosted Decision Tree Regression algorithm** will forecast a numeric value, not make a prediction between several categories. The **K-means algorithm** will separate similar data points into groups, not make a prediction between several categories.

Azure Machine Learning designer is a no code, drag and drop interface. The **Azure Machine Learning SDK** is a code only interface in Python. The **Azure CLI** is a code only interface and does not support all the features of Azure Machine Learning. **Azure Machine Learning studio** is a web-based tool for managing an Azure Machine Learning instance.

The **regression and the classification algorithms** are supported by automated machine learning. The **dimensionality reduction and the clustering algorithms** not supported by automated machine learning.

Enabling featurization will make sure AutoML tries to apply the encoding transformation. **Enabling transform and encode** is a technique that will be tried when activating featurization but cannot be enabled on its own. **Feature scaling and normalization** is enabled by default. **Impute missing values** is a technique that will be tried when activating featurization but cannot be enabled on its own.

The **RandomParameterSampling sampling** strategy is faster than GridParameterSampling because it randomly chooses combinations of parameters in the search space to try, rather than trying them all. The **GridParameterSampling sampling** strategy takes the maximum amount of time to identify good values since it tries each possible combination in the search space. The **BanditPolicy** is a technique for early termination of hyperparameter searching. The **TruncationSelectionPolicy** is a technique for early termination of hyperparameter searching. BanditPolicy and TruncationSelectionPolicy are not sampling techniques.

The **QNormal distribution** supports discrete hyperparameters. The **lognormal, uniform, and LogUniform distributions** do not support discrete hyperparameters.

A **compute instance** is a single compute node that process the data much faster. **Numpy** is a python library for operating on arrays, not tabular data. A **Spark component** in a pipeline job would speed up the processing but would require more overhead and setup than a **stand-alone Spark job**. Therefore, the stand-alone Spark job is the best option for wrangling this data at scale with the least setup and overhead.

AzureFileDatastore class is used to access data in an Azure File Datastore. The **AzureDataLakeGen2** class is used to access data in Azure Data Lake Gen 2. The **Data** class is used to access data that has already been processed for training and scoring models. Only the **AzureBlobDatastore** class is used to access data in an Azure Blob datastore.

The **AUC** is a good metric to evaluate the performance of a classification model. The other metrics are used for regressions, not classifications.

An Execute Python Script component must implement an **azureml_main** method. A Create Python Model component must implement a **train** and **predict** methods. The **load_data** method is used to construct a data object but is not part of the Execute Python Script component.

Converting the training pipeline to an inference pipeline is necessary before being able to deploy. **Ensuring the performance of the model**, although desirable, is not strictly necessary. **Executing a Python Script** is a matter of choice when designing the pipeline. Finally, **sending data** to the URI is only possible after the model has been deployed.

Azure Databricks is a compute resource, but it is not supported by Azure Machine Learning designer. **Azure Container Registry** is an Azure service for saving and reuse of docker containers. **Azure Machine Learning Compute** is a compute resource supported by Azure Machine Learning designer, but only for training purposes. Only **Azure Kubernetes Service** is supported by Azure Machine Learning designer and used for real time endpoint deployment.

The **Extract N-Gram Features** from Text component and the Convert Word to Vector component transforms input text into features for making predictions on. The **Split Data component** is for splitting data into two parts: train and test sets. Only the Preprocess Text component removes stop words from input text.

To get the boundaries of the different entities that are present in an image using Azure AutoML, the **Object Detection Computer Vision model** must be used.

The **Azure Machine Learning Python SDKv2, the Azure CLI v2, and the Jupyter Notebook feature** in Azure Machine Learning Studio are all code first tools. The **AutoML feature in Azure Machine Learning**

Studio is a low/no code GUI-based tool for automatically training and optimizing machine learning models.

When using sklearn, the **score method** will return the mean accuracy on the given test data and labels. The **fit, predict, and log_metric methods** do not return the accuracy, but are useful to respectively train the model, get the predictions, and log the accuracy once computed.

the **get_run method** will return the metadata, including metrics, for a given run. The other options perform unrelated operations on the run.

To cover all the combinations of the hyperparameters, **grid search** is the only sampling technique that works, and **it supports discrete hyperparameters** only.