akride Documentation

Release 0.2.12

Akridata Inc.

Nov 21, 2023

CONTENTS

Python SDK Client to interact with Akridata Data Explorer

CHAPTER

ONE

CONTENTS

1.1 Data Explorer Client SDK

1.1.1 Introduction

`akride` is a Python SDK that facilitates interaction with Akridata's Data Explorer, providing seamless integration and tools for streamlined data analysis.

Akridata's Data Explorer allows you to save time building the best training and test sets for your application.

`akride` will help you:

- Visualize your data
- Detect and remove outliers
- Check class imbalance
- Sample the data and remove duplications
- Apply Image-based-Search
- Analyze model's accuracy

and much more!

1.1.2 Installation

You can install the latest stable version of `akride` via `pip` -

for a CPU-only version:

```
pip install akride[cpu]
```

or on a GPU-enabled machine:

```
pip install akride[gpu]
```

1.1.3 QuickStart

Start using Akride by copying the following code snippet into your Python terminal:

```
from akride import AkriDEClient
SAAS_ENDPOINT="https://app.akridata.ai"
API_KEY=<your-api-key>
# API Key Configurations
sdk_config_dict = {
  "saas_endpoint": SAAS_ENDPOINT,
  "api_key": API_KEY,
  "mode": "saas"
}


client = AkriDEClient(sdk_config_dict=sdk_config_dict)
print(client.get_server_version())
```

1.1.4 Documentation

For detailed documentation on how to use the akride client and its capabilities, please refer to the official SDK documentation.

For more information about Akridata's Data Explorer, please refer to the official Akridata documentation.

1.1.5 Examples

Check out the akride-examples repository for examples of using `akride` client to interact with DataExplorer

---

For more information about AkriData, please visit akridata.ai.

1.2 License

```
Copyright (C) 2023, Akridata, Inc - All Rights Reserved.
Unauthorized copying of this file, via any medium is strictly prohibited
```

1.3 akride

1.3.1 akride package

Subpackages

akride.core package

Subpackages

akride.core.conf package

Module contents

akride.core.entities package

Submodules

akride.core.entities.catalogs module

class akride.core.entities.catalogs.**Catalog**(*info: GetCatalogResponse*)

Bases: *Entity*

Class representing a catalog entity.

delete() → None

Deletes an entity.

Return type
None

akride.core.entities.datasets module

class akride.core.entities.datasets.**Dataset**(*info: DataSetsItem*)

Bases: *Entity*

Class representing a dataset entity.

delete() → None

Deletes an entity.

Return type
None

akride.core.entities.entity module

class akride.core.entities.entity.**Entity**(*entity_id*, *name*)

Bases: ABC

Abstract base class representing an entity in the system.

abstract delete() → None

Deletes an entity.

Return type
None

`get_id`() → str | None

Method for getting the ID of the entity.

Returns
The ID of the entity.

Return type
str

`get_name`() → str | None

Method for getting the name of the entity.

Returns
The name of the entity.

Return type
str

`to_dict`() → dict

Method for converting the entity to a dictionary.

Returns
A dictionary representing the entity.

Return type
dict

akride.core.entities.jobs module

`class akride.core.entities.jobs.Job`(*info: CreateJobRequestResponse*)

Bases: *Entity*

Class representing a job entity.

`property dataset_id`

`delete`() → None

Deletes an entity.

Return type
None

`property pipeline_id`

`class akride.core.entities.jobs.JobSpec`(*dataset:* Dataset, ***kwargs*)

Bases: Dict

Class representing a job specification. TODO: separate specs for different job types

akride.core.entities.pipeline module

class akride.core.entities.pipeline.**Pipeline**(*info: Pipeline*)

Bases: *Entity*

Class representing a Pipeline entity.

delete() → None

Deletes an entity.

Return type
None

akride.core.entities.resultsets module

class akride.core.entities.resultsets.**Resultset**(*info: ResultsetListResponseItem*)

Bases: *Entity*

Class representing a result set entity.

delete() → None

Deletes an entity.

Return type
None

property job_id

property version

Module contents

akride.core.models package

Submodules

akride.core.models.catalog_details module

class akride.core.models.catalog_details.**CatalogDetails**(*table_name: str*, *catalog_csv_file: str*)

Bases: object

Class representing parameters details for creating a catalog.

table_name

str The name of the table to create for the catalog.

Type
str

catalog_csv_file

str The path to the CSV file containing new catalog data.

Type
str

catalog_csv_file: *str*

table_name: *str*

akride.core.models.progress_info module

class akride.core.models.progress_info.**ProgressInfo**(*percent_completed: float*, *message: str | NoneType = None*, *completed: bool = False*)

Bases: object

completed: *bool* = False

message: *str | None* = None

percent_completed: *float*

Module contents

Submodules

akride.core.constants module

class akride.core.constants.**Constants**

Bases: object

BLOB_TABLE_COLUMNS = ['partition_start', 'partition_end', 'workflow_id', 'session_id', 'blob_id']

DATASET_FILES_COLUMNS = ['partition_start', 'partition_end', 'workflow_id', 'session_id', 'file_path']

DEBUGGING_ENABLED = False

FILE_TYPES = ['blobs', 'coreset', 'projections', 'sketch', 'thumbnail']

INGEST_FILES_COUNT_IN_ONE_PARTITION = 10000

INGEST_WF_TOKEN_SIZE = 1024

LOG_CONFIG_FILE_NAME = 'pylogconf.yaml'

PARTITIONED_TABLE_COLUMNS = ['partition_start', 'partition_end', 'workflow_id', 'session_id', 'file_path', 'file_id', 'partition_id']

PARTITION_SIZE = 300000000

```
PRIMARY_TABLE_COLUMNS = ['partition_start', 'partition_end', 'workflow_id',
'session_id', 'frame_idx_in_blob', 'blob_idx_in_partition', 'file_path',
'timestamp', 'file_id', 'frame_idx_in_file', 'file_name', 'total_frames_in_file']

PROCESS_WF_TOKEN_SIZE = 1500

SUMMARY_TABLE_COLUMNS = ['partition_start', 'partition_end', 'workflow_id',
'session_id', 'coreset', 'projections', 'sketch', 'thumbnail']

THUMBNAIL_AGGREGATOR_SDK_DETAILS = {'class_name': 'ThumbnailAggregator',
'cleanup_method': None, 'init_method': None, 'init_params': None, 'module':
'pyakri_de_filters.thumbnail.thumbnail_aggregator', 'run_method': 'run'}
```

akride.core.enums module

class akride.core.enums.**BackgroundTaskType**(*value*)

Bases: Enum

Specifies the type of background task

DATASET_INGESTION = 'dataset_ingestion'

class akride.core.enums.**CatalogTableType**(*value*)

Bases: Enum

TableType for create view

EXTERNAL = 'external'

INTERNAL = 'internal'

class akride.core.enums.**ClusterAlgoType**(*local_vars_configuration=None*)

Bases: ClusterAlgorithms

Cluster algorithms supported by DataExplorer

class akride.core.enums.**DataType**(*value*)

Bases: Enum

Supported Data types

IMAGE = 'image/*'

VIDEO = 'video/*'

class akride.core.enums.**DatastoreType**(*value*)

Bases: Enum

Supported datastore types

AZURE = 2

GCS = 3

LOCAL = 0

S3 = 1

class akride.core.enums.**EmbedAlgoType**(*local_vars_configuration=None*)

Bases: EmbeddingAlgorithms

Embedding algorithms supported by DataExplorer

class akride.core.enums.**FeaturizerType**(*value*)

Bases: Enum

Type of featurizer to be used for ingestion FULL_IMAGE: Features generated on the full image PATCH: Features generated on a grid of cells over image. Supports patch search EXTERNAL: Features are generated externally and registered against dataset

EXTERNAL = 2

FULL_IMAGE = 0

PATCH = 1

class akride.core.enums.**JobContext**(*value*)

Bases: Enum

Specifies the context that samples are requested under

CLUSTER_RETRIEVAL = 2

CONFUSION_MATRIX_CELL = 0

CORESET_SAMPLING = 3

SIMILARITY_SEARCH = 1

class akride.core.enums.**JobStatisticsContext**(*value*)

Bases: Enum

Specifies the type of statistics to be retrieved

CONFIDENCE_HISTOGRAM = 2

CONFUSION_MATRIX = 0

PRECISION_RECALL_CURVE = 1

class akride.core.enums.**JobType**(*local_vars_configuration=None*)

Bases: JobType

Supported Job types

classmethod is_analyze_job(*job_type*) → bool

akride.core.exceptions module

exception akride.core.exceptions.**BaseError**(*message: str*)

Bases: Exception

Base class for creating custom exception classes.

message

Information about the error that occurred.

Type
str

class akride.core.exceptions.**ErrorMessages**

Bases: object

Class that that holds all error messages used in the sdk

SDK_SERVER_ERR_01_NOT_REACHABLE = 'Server not reachable'

SDK_USER_ERR_01_INVALID_AUTH = 'Invalid Authentication Config'

exception akride.core.exceptions.**InvalidAuthConfigError**(*message: str*)

Bases: *UserError*

Custom exception class defined to handle errors raised due to Invalid api-key

exception akride.core.exceptions.**ServerError**(*message: str*)

Bases: *BaseError*

Custom exception class for handling errors that occur in server-related operations. This will capture all 5xx errors

exception akride.core.exceptions.**ServerNotReachableError**(*message: str*)

Bases: *BaseError*

Error thrown when the client is unable to connect to the server

exception akride.core.exceptions.**UserError**(*message: str*)

Bases: *BaseError*

Custom exception class for user-defined errors. This will capture all 4xx errors

akride.core.types module

class akride.core.types.**AnalyzeJobParams**(*catalog_config: CatalogDetails*, *plot_featurizer: str = 'content'*, *confidence_config: List[float] = [0, 0.2, 0.4, 0.6, 0.8, 1]*, *iou_config: List[float] = [0.1, 0.3, 0.5, 0.7, 0.9, 1]*)

Bases: object

catalog_config: *CatalogDetails*

confidence_config: List[float]

iou_config: List[float]

plot_featurizer: str

class akride.core.types.**CatalogDetails**(*score_column: str | None = None*, *ground_truth_class_column: str | None = None*, *prediction_class_column: str | None = None*, *ground_truth_coordinates_column: str | None = None*, *prediction_coordinates_column: str | None = None*, *ground_truth_coordinates_class_column: str | None = None*, *prediction_coordinates_class_score_column: str | None = None*)

Bases: object

ground_truth_class_column: str

ground_truth_coordinates_class_column: str

ground_truth_coordinates_column: str

prediction_class_column: str

prediction_coordinates_class_score_column: str

prediction_coordinates_column: str

score_column: str

class akride.core.types.**CatalogTable**(*table_name: str*, *table_type:* CatalogTableType | *None = None*, *pipeline_id: str | None = None*, *catalog_name: str | None = None*, *schema_name: str | None = None*, *alias_name: str | None = None*, *is_view: bool | None = False*)

Bases: object

class akride.core.types.**ClientManager**(*am_client: ApiClient*, *dsp_client: ApiClient*, *background_task_manager:* BackgroundTaskManager)

Bases: object

Dataclass managing different APIClient required to connect with DataExplorer services

am_client: ApiClient

background_task_manager: *BackgroundTaskManager*

dsp_client: ApiClient

class akride.core.types.**ClusterRetrievalSpec**(***kwargs*)

Bases: *JobOpSpec*

Class representing a cluster retrieval specification.

class akride.core.types.**Column**(*name: str*, *type: str*)

Bases: object

Type representing a Table Column

name: str

type: str

class akride.core.types.**ConfusionMatrix**(*data*, *labels*)

Bases: *JobStatistics*

Class representing a confusion matrix.

to_dict() → dict

Method for converting this object to a dictionary.

Returns
A dictionary representing this object.

Return type
dict

class akride.core.types.**ConfusionMatrixCellSpec**(***kwargs*)

Bases: *JobOpSpec*

Class representing a confusion matrix cell specification.

class akride.core.types.**CoresetSamplingSpec**(***kwargs*)

Bases: *JobOpSpec*

Class representing a coreset sampling specification.

class akride.core.types.**JobOpSpec**(***kwargs*)

Bases: Dict

class akride.core.types.**JobStatistics**

Bases: object

class akride.core.types.**JoinCondition**(*left_column*, *right_column*)

Bases: object

class akride.core.types.**PlotFeaturizer**(*local_vars_configuration=None*)

Bases: PlotFeaturizer

PlotFeaturizer for Analyze job

class akride.core.types.**SampleInfoList**(*job_id: str = ''*, *point_ids: List | None = None*)

Bases: object

Class representing a list of samples.

append_sample(*sample: ResultsetResponseFrameItem*)

get_fullres_urls()

get_local_paths()

get_point_ids()

get_thumbnail_urls()

to_dict() → dict

Method for converting this object to a dictionary.

Returns
A dictionary representing the this object.

Return type
dict

class akride.core.types.**SimilaritySearchSpec**(***kwargs*)

Bases: *JobOpSpec*

Class representing a similarity search specification.

Module contents

Submodules

akride.background_task_manager module

class akride.background_task_manager.**BackgroundTaskManager**

Bases: object

Helper class to manage background task

is_task_running(*entity_id: str*, *task_type:* BackgroundTaskType) → bool

:param : :type : param entity_id: Entity ID associated with the task. :param : :type : param task_type: The type of the background task.

Returns
a boolean representing whether task is running or not.

Return type
Boolean

start_task(*entity_id: str*, *task_type:* BackgroundTaskType, *target_function*, **args*, ***kwargs*) → BackgroundTask

Start a background task.

:param : :type : param task_type: The type of the background task. :param : :type : param entity_id: Entity ID associated with the task :param : :type : param target_function: The target function to run :param : :type : param args: Arguments for the target function :param : :type : param kwargs: Keyword arguments for the target function

Returns
background task object

Return type
BackgroundTask

akride.client module

Copyright (C) 2023, Akridata, Inc - All Rights Reserved. Unauthorized copying of this file, via any medium is strictly prohibited

class akride.client.**AkriDEClient**(*sdk_config_tuple: Tuple[str, str] | None = None*, *sdk_config_dict: dict | None = None*, *sdk_config_file: str | None = None*)

Bases: object

Client class to connect to DataExplorer

add_to_catalog(*dataset:* Dataset, *table_name: str*, *csv_file_path: str*) → bool

Adds new items to an existing catalog.

Parameters

- **dataset** (Dataset) – The dataset to import the catalog into.

- **table_name** (*str*) – The name of the table to create for the catalog.
- **csv_file_path** (*str*) – The path to the CSV file containing new catalog data.

Returns
Indicates whether the operation was successful.

Return type
bool

create_dataset(*spec: Dict[str, Any]*) → *Entity*

Creates a new dataset entity.

Parameters
spec (*Dict[str, Any]*) –

The dataset spec. The spec should have the following fields:

dataset_name
[str] The name of the new dataset.

dataset_namespace
[str, optional] The namespace for the dataset, by default 'default'.

data_type
[DataType, optional] The type of data to store in the dataset, by default DataType.IMAGE.

glob_pattern
[str, optional] The glob pattern for the dataset, by default '*(png|jpg|gif|jpeg|tiff|tif|bmp)'.

overwrite
[bool, optional] Overwrite if a dataset with the same name exists.

Returns
The created entity

Return type
Entity

create_job(*spec:* JobSpec) → *Job*

Creates an explore job for the specified dataset.

Parameters:

dataset: Dataset
The dataset to explore.

spec: JobSpec
The job specification.

Returns:

Job
The newly created Job object.

`create_job_spec`(*dataset: Dataset*, *job_type: str | JobType = 'EXPLORE'*, *job_name: str = ''*, *predictions_file: str = ''*, *cluster_algo: str | ClusterAlgoType = 'hdbscan'*, *embed_algo: str | EmbedAlgoType = 'umap'*, *num_clusters: int | None = None*, *max_images: int = 1000*, *catalog_table: CatalogTable | None = None*, *analyze_params: AnalyzeJobParams | None = None*, *pipeline: Pipeline | None = None*, *filters: List[Condition] | None = None*) → *JobSpec*

Creates a JobSpec object that specifies how a job is to be created.

Parameters:

dataset: Dataset
The dataset to explore.

job_type
[JobType, optional] The job type

job_name
[str, optional] The name of the job to create. A unique name will be generated if this is not given.

predictions_file: str, optional

The path to the catalog file containing predictions and ground truth. This file must be formatted according to the specification at:

https://docs.akridata.ai/docs/analyze-job-creation-and-visualization

cluster_algo
[ClusterAlgoType, optional] The clustering algorithm to use.

embed_algo
[EmbedAlgoType, optional] The embedding algorithm to use.

num_clusters
[int, optional] The number of clusters to create.

max_images
[int, optional] The maximum number of images to use.

catalog_table: CatalogTable, optional
The catalog to be used for creating this explore job. This defaults to the internal primary catalog that is created automatically when a dataset is created. default: "primary"

analyze_params: AnalyzeJobParams, optional
Analyze job related configuration parameters

filters
[List[Condition], optional] The filters to be used to select a subset of samples for this job. These filters are applied to the catalog specified by catalog_name.

`create_resultset`(*spec: Dict[str, Any]*) → *Entity*

Creates a new resultset entity.

Parameters
spec (*Dict[str, Any]*) –

The resultset spec. The spec should have the following fields:

job: Job
The associated job object.

name
[str] The name of the new resultset.

samples: SampleInfoList
The samples to be included in this resultset.

Returns
The created entity

Return type
Entity

create_view(*view_name: str*, *description: str | None*, *dataset:* Dataset, *left_table:* CatalogTable, *right_table:* CatalogTable, *join_condition:* JoinCondition) → str

Create a SQL view for visualization

Parameters

- **view_name** (*str*) – Name of the view to create
- **description** (*Optional[str]*) – Description text
- **dataset** (Dataset) – Dataset object
- **left_table** (*TableInfo*) – Left Table of the create view query
- **right_table** (*TableInfo*) – Right Table of the create view query
- **join_condition** (JoinCondition) – JoinCondition which includes the
- **table** (*column from the left and the right*) –

Returns
view id

Return type
str

delete_catalog(*catalog:* Catalog) → bool

Deletes a catalog object.

Parameters
catalog (Catalog) – The catalog object to delete.

Returns
Indicates whether the operation was successful.

Return type
bool

delete_dataset(*dataset:* Dataset) → bool

Deletes a dataset object.

Parameters
dataset (Dataset) – The dataset object to delete.

Returns
Indicates whether this entity was successfully deleted

Return type
bool

delete_job(*job:* Job) → bool

Deletes a job object.

Parameters
job (Job) – The job object to delete.

Returns
Indicates whether the operation was successful.

Return type
bool

delete_resultset(*resultset:* Resultset) → bool

Deletes a resultset object.

Parameters
resultset (Resultset) – The resultset object to delete.

Returns
Indicates whether the operation was successful.

Return type
bool

get_all_columns(*dataset:* Dataset, *table:* CatalogTable) → List[*Column*]

Returns all columns for a table/view

Parameters

- **dataset** (Dataset) – Dataset object
- **table** (*TableInfo*) – Table Information

Returns
List of columns of the table

Return type
List[*Column*]

get_attached_pipelines(*dataset:* Dataset, *version:* *str* | *None* = *None*) → List[*Pipeline*]

Get pipelines attached for dataset given a dataset version

Parameters

- **dataset** (Dataset) – Dataset object
- **version** (*str, optional*) – Dataset version. Defaults to None in which
- **used** (*case the latest version would be*) –

Returns
List of pipelines attached with the dataset

Return type
List[*Pipeline*]

get_catalog_by_name(*dataset:* Dataset, *name:* *str*) → *Entity* | None

Retrieves a catalog with the given name.

Parameters

- **dataset** (Dataset) – The dataset to retrieve the catalog from.

- **name** (*str*) – The name of the catalog to retrieve.

Returns
The Entity object representing the catalog.

Return type
Entity

get_catalog_tags(*samples:* SampleInfoList) → DataFrame

Retrieves the catalog tags corresponding to the given samples.

Parameters
samples (SampleInfoList) – The samples to retrieve catalog tags for.

Returns
A dataframe of catalog tags.

Return type
pd.DataFrame

get_catalogs(*attributes: Dict[str, Any] = {}*) → List[*Entity*]

Retrieves information about catalogs that have the given attributes.

Parameters
attributes (*Dict[str, Any]*) – The filter specification. It may have the following optional fields:

name
[str] filter by catalog name

status
[str] filter by catalog status, can be one of "active","inactive", "refreshing", "offline", "invalid-config"

Returns
A list of Entity objects representing catalogs.

Return type
List[*Entity*]

get_dataset_by_name(*name: str*) → *Entity* | None

Retrieves a dataset with the given name.

Parameters
name (*str*) – The name of the dataset to retrieve.

Returns
The Entity object representing the dataset.

Return type
Entity

get_datasets(*attributes: Dict[str, Any] = {}*) → List[*Entity*]

Retrieves information about datasets that have the given attributes.

Parameters
attributes (*Dict[str, Any], optional*) – The filter specification. It may have the following optional fields:

search_key
[str] Filter across fields like dataset id, and dataset name.

Returns
A list of Entity objects representing datasets.

Return type
List[*Entity*]

get_fullres_image_urls(*samples:* SampleInfoList) → Dict

Retrieves the full-resolution image urls for the give samples.

Parameters
samples (SampleInfoList) – The samples to retrieve full res image urls for.

Returns
A dictionary containing the full-resolution image URLs for each sample.

Return type
Dict

get_fullres_images(*samples:* SampleInfoList) → List[Image]

Retrieves the full-resolution images for the provided job.

Parameters
samples (SampleInfoList) – The samples to retrieve images for.

Returns
A list of images.

Return type
List[Image.Image]

get_job_by_name(*name: str*) → *Job*

Retrieves a job with the given name.

Parameters
name (*str*) – The name of the job to retrieve.

Returns
The Entity object representing the job.

Return type
Entity

get_job_display_panel(*job:* Job) → str

Retrieves the job panel URI the Data Explorer.

Parameters
job (Job) – The Job object to be queried.

Returns
The job panel URL.

Return type
str

get_job_samples(*job:* Job, *job_context:* JobContext, *spec:* SimilaritySearchSpec | ConfusionMatrixCellSpec | ClusterRetrievalSpec | CoresetSamplingSpec, ***kwargs*) → *SampleInfoList*

Retrieves the samples according to the given specification.

Parameters

- **job** (Job) – The Job object to get samples for.

- **job_context** (JobContext) – The context in which the samples are requested for.
- **spec** (*Union[*) – SimilaritySearchSpec, ConfusionMatrixCellSpec, ClusterRetrievalSpec, CoresetSamplingSpec
- **]** – The job context spec.
- ****kwargs** (*Additional keyword arguments*) –
- **arguments** (*Supported keyword*) –

  iou_config_threshold: float, optional
  Threshold value for iou config

  confidence_score_threshold: float, optional
  Threshold value for confidence score

Returns
A SampleInfoList object.

Return type
SampleInfoList

get_job_samples_from_file_path(*job:* Job, *file_info:* *List[str]*) → Dict

Retrieves the samples according to the given specification.

Parameters

- **job** (Job) – The Job object to get samples for. The job context spec.
- **file_info** (*List[str]*) – List of file_paths for the images of interest

Returns
dictionary of map between file_path and point_ids

Return type
Dict

get_job_statistics(*job:* Job, *context:* JobStatisticsContext, ***kwargs*) → *JobStatistics*

Retrieves statistics info from an analyze job.

Parameters

- **job** (Job) – The Job object to get statistics for.
- **context** (JobStatisticsContext) – The type of statistics to retrieve.
- ****kwargs** (*Additional keyword arguments*) –
- **arguments** (*Supported keyword*) –

  iou_config_threshold: float, optional
  Threshold value for iou config

  confidence_score_threshold: float, optional
  Threshold value for confidence score

Returns
A job statistics object.

Return type
JobStatistics

get_jobs(*attributes: Dict[str, Any] = {}*) → List[*Entity*]

Retrieves information about jobs that have the given attributes.

Parameters
attributes (*Dict[str, Any]*) – The filter specification. It may have the following optional fields:

data_type
[str] The data type to filter on. This can be 'IMAGE' or 'VIDEO'.

job_type
[str] The job type to filter on - 'EXPLORE', 'ANALYZE' etc.

search_key
[str] Filter jobs across fields like job name, dataset id, and dataset name.

Returns
A list of Entity objects representing jobs.

Return type
List[*Entity*]

get_progress_info(*task: BackgroundTask*) → *ProgressInfo*

Gets the progress of the specified task.

Parameters
task (*BackgroundTask*) – The task object to retrieve the progress information for.

Returns
The progress information

Return type
ProgressInfo

get_resultset_by_name(*name: str*) → *Entity* | None

Retrieves a resultset with the given name.

Parameters
name (*str*) – The name of the resultset to retrieve.

Returns
The Entity object representing the resultset.

Return type
Entity

get_resultset_samples(*resultset:* Resultset) → *SampleInfoList*

Retrieves the samples of a resultset

Parameters
resultset (Resultset) – The Resultset object to get samples for.

Returns
A SampleInfoList object.

Return type
SampleInfoList

get_resultsets(*attributes: Dict[str, Any] = {}*) → List[*Entity*]

Retrieves information about resultsets that have the given attributes.

Parameters
attributes (*Dict[str, Any], optional*) – The filter specification. It may have the following optional fields:

search_key
[str] Filter across fields like dataset id, and dataset name.

Returns
A list of Entity objects representing resultsets.

Return type
List[*Entity*]

get_server_version() → str

Get Dataexplorer server version

Returns
server version

Return type
str

get_thumbnail_images(*samples:* SampleInfoList) → List[Image]

Retrieves the thumbnail images corresponding to the samples.

Parameters
samples (SampleInfoList) – The samples to retrieve thumbnails for.

Returns
A list of thumbnail images.

Return type
List[Image.Image]

import_catalog(*dataset:* Dataset, *table_name: str*, *csv_file_path: str*, *create_view: bool = True*, *file_name_column: str | None = None*, *pipeline_name: str | None = None*) → bool

Method for importing an external catalog into a dataset.

Parameters

- **dataset** (Dataset) – The dataset to import the catalog into.
- **table_name** (*str*) – The name of the table to create for the catalog.
- **csv_file_path** (*str*) – The path to the CSV file containing the catalog data.
- **create_view** (*bool default: True*) – Create a view with imported catalog and primary catalog table
- **file_name_column** (*str*) – Name of the column in the csv file that contains the absolute filename
- **pipeline_name** (*str*) – Name of pipeline whose primary table will be joined with the imported table. Ignored if create_view is false

Returns
Indicates whether the operation was successful.

Return type
bool

ingest_dataset(*dataset:* Dataset, *data_directory: str*, *use_patch_featurizer: bool = True*, *async_req: bool = False*, *catalog_details:* CatalogDetails | *None = None*) → BackgroundTask | None

Starts an asynchronous ingest task for the specified dataset.

Parameters

- **dataset** (*Dataset*) – The dataset to ingest.
- **data_directory** (*str*) – The path to the directory containing the dataset files.
- **use_patch_featurizer** (*bool, optional*) – Ingest dataset to enable patch-based similarity searches.
- **async_req** (*bool, optional*) – Whether to execute the request asynchronously.
- **catalog_details** (*Optional[CatalogDetails]*) – Parameters details for creating a catalog

Returns
A task object

Return type
BackgroundTask

update_resultset(*resultset:* Resultset, *add_list:* SampleInfoList | *None = None*, *del_list:* SampleInfoList | *None = None*) → bool

Updates a resultset.

Parameters

- **resultset** (*Resultset*) – The resultset to be updated.
- **add_list** (*SampleInfoList, optional*) – The list of samples to be added.
- **del_list** (*SampleInfoList, optional*) – The list of samples to be deleted.

Returns
Indicates whether the operation was successful.

Return type
bool

wait_for_completion(*task: BackgroundTask*) → *ProgressInfo*

Waits for the specified task to complete.

Parameters
task (*BackgroundTask*) – The ID of the job to wait for.

Returns
The progress information

Return type
ProgressInfo

Module contents

akride.**init**(*sdk_config_tuple: Tuple[str, str] | None = None*, *sdk_config_dict: dict | None = None*, *sdk_config_file: str | None = ''*) → *AkriDEClient*

Initializes the AkriDEClient with the saas_endpoint and api_key values The init params could be passed in different ways, incase multiple options are used to pass the init params the order of preference would be 1. sdk_config_tuple, 2. sdk_config 3. sdk_config_file

Get the config by signing in to Data Explorer UI and navigating to Utilities → Get CLI/SDK config :param sdk_config_tuple: A tuple consisting of saas_endpoint and api_key in that order :type sdk_config_tuple: tuple :param sdk_config_dict: dictionary containing "saas_endpoint" and "api_key" :type sdk_config_dict: dict :param sdk_config_file: Path to the the SDK config file downloaded from Dataexplorer :type sdk_config_file: str

Raises

- ***InvalidAuthConfigError*** – if api-key/host is invalid:
- ***ServerNotReachableError*** – if the server is unreachable:

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

INDEX

D

E

F

G

I

J

L

M

N

P

R

S

T

U

V

W