Experiment Notebook - AutoAI Notebook v2.1.7

This notebook contains the steps and code to demonstrate support of AutoAI experiments in the watsonx.ai Runtime. It introduces Python API commands for data retrieval, training experiments, persisting pipelines, testing pipelines, refining pipelines, and scoring the resulting model.

Note: Notebook code generated using AutoAI will execute successfully. If code is modified or reordered, there is no guarantee it will successfully execute. For details, see: Saving an Auto AI experiment as a notebook

Some familiarity with Python is helpful. This notebook uses Python 3.11 and the `ibm-watsonx-ai` package.

Notebook goals

The learning goals of this notebook are:

- Defining an AutoAI experiment
- Training AutoAI models
- Comparing trained models
- Deploying the model as a web service
- Scoring the model to generate predictions

Contents

This notebook contains the following parts:

Setup

Package installation

Before you use the sample code in this notebook, install the following packages:

- ibm-watsonx-ai,
- autoai-libs,
- lale,
- scikit-learn,
- xgboost,
- lightgbm,
- snapml

In [ ]:
```
!pip install ibm-watsonx-ai | tail -n 1
!pip install autoai-libs~=2.0 | tail -n 1
!pip install -U 'lale~=0.8.3' | tail -n 1
!pip install scikit-learn==1.3.* | tail -n 1
!pip install xgboost==2.0.* | tail -n 1
!pip install lightgbm==4.2.* | tail -n 1
!pip install snapml==1.14.* | tail -n 1
```

Experiment configuration

Experiment metadata

This cell defines the metadata for the experiment, including: training_data_references, training_result_reference, experiment_metadata.

In [ ]:
```
from ibm_watsonx_ai.helpers import DataConnection
from ibm_watsonx_ai.helpers import ContainerLocation

training_data_references = [
    DataConnection(
        data_asset_id='44785a94-5108-4bdc-974a-9481fea449e8'
    ),
]
training_result_reference = DataConnection(
    location=ContainerLocation(
        path='auto_ml/a98f60ba-cd6c-44b2-8c21-f421bf393d86/wml_data/596e6f61-d075-4214-
        model_location='auto_ml/a98f60ba-cd6c-44b2-8c21-f421bf393d86/wml_data/596e6f61-
        training_status='auto_ml/a98f60ba-cd6c-44b2-8c21-f421bf393d86/wml_data/596e6f61
```

```
    )
)
```

In [ ]:
```
experiment_metadata = dict(
    prediction_type='multiclass',
    prediction_column='Fault Type',
    holdout_size=0.1,
    scoring='accuracy',
    csv_separator=',',
    random_state=33,
    max_number_of_estimators=2,
    training_data_references=training_data_references,
    training_result_reference=training_result_reference,
    deployment_url='https://us-south.ml.cloud.ibm.com',
    project_id='4ea368dd-fbf7-42b9-bbfa-309cf438afe3',
    drop_duplicates=True,
    include_batched_ensemble_estimators=['BatchedTreeEnsembleClassifier(ExtraTreesClass
    feature_selector_mode='auto'
)
```

watsonx.ai connection

This cell defines the credentials required to work with the watsonx.ai Runtime.

Action: Provide the IBM Cloud apikey, For details, see documentation.

In [ ]:
```
import getpass

api_key = getpass.getpass("Please enter your api key (press enter): ")
```

In [ ]:
```
from ibm_watsonx_ai import Credentials

credentials = Credentials(
    api_key=api_key,
    url=experiment_metadata['deployment_url']
)
```

Working with completed AutoAI experiment

This cell imports the pipelines generated for the experiment. The best pipeline will be saved as a model.

Get fitted AutoAI optimizer

In [ ]:
```
from ibm_watsonx_ai.experiment import AutoAI

pipeline_optimizer = AutoAI(credentials, project_id=experiment_metadata['project_id']).
```

Use `get_params()` to retrieve configuration parameters.

In [ ]:
```
pipeline_optimizer.get_params()
```

Pipelines comparison

Use the `summary()` method to list trained pipelines and evaluation metrics information in the form of a Pandas DataFrame. You can use the DataFrame to compare all discovered pipelines and select the one you like for further testing.

In [ ]:
```
summary = pipeline_optimizer.summary()
best_pipeline_name = list(summary.index)[0]
summary
```

Get pipeline as a scikit-learn pipeline model

After you compare the pipelines, download and save a scikit-learn pipeline model object from the AutoAI training job.

Tip: To get a specific pipeline, pass the pipeline name in:

```
pipeline_optimizer.get_pipeline(pipeline_name=pipeline_name)
```

In [ ]:
```
pipeline_model = pipeline_optimizer.get_pipeline()
```

Next, check the importance of features for selected pipeline.

In [ ]:
```
pipeline_optimizer.get_pipeline_details()['features_importance']
```

Tip: If you want to check all the details of the model evaluation metrics, use:

```
pipeline_optimizer.get_pipeline_details()
```

Score the fitted pipeline with the generated scorer using the holdout dataset.

1. Get sklearn pipeline_model

In [ ]:
```
sklearn_pipeline_model = pipeline_optimizer.get_pipeline(astype=AutoAI.PipelineTypes.SK
```

2. Get training and testing data

In [ ]:
```
from ibm_watsonx_ai import APIClient

client = APIClient(credentials=credentials)

if 'space_id' in experiment_metadata:
    client.set.default_space(experiment_metadata['space_id'])
else:
    client.set.default_project(experiment_metadata['project_id'])
```

```
training_data_references[0].set_client(client)
```

In [ ]:
```
_, X_test, _, y_test = training_data_references[0].read(experiment_metadata=experiment_
```

3. Define scorer, score the fitted pipeline with the generated scorer using the holdout dataset.

In [ ]:
```
from sklearn.metrics import get_scorer

scorer = get_scorer(experiment_metadata['scoring'])
```

In [ ]:
```
score = scorer(sklearn_pipeline_model, X_test.values, y_test.values)
print(score)
```

Inspect pipeline

Visualize pipeline model

Preview pipeline model stages as a graph. Each node's name links to a detailed description of the stage.

In [ ]:
```
pipeline_model.visualize()
```

Preview pipeline model as a Python code

In the next cell, you can preview the saved pipeline model as a Python code.
You can review the exact steps used to create the model.

Note: If you want to get sklearn representation, add the following parameter to the `pretty_print` call: `astype='sklearn'` .

In [ ]:
```
pipeline_model.pretty_print(combinators=False, ipython_display=True)
```

Calling the `predict` method

If you want to get a prediction by using the pipeline model object, call `pipeline_model.predict()` .

Note: If you want to work with a pure sklearn model:

- add the following parameter to the `get_pipeline` call: `astype='sklearn'` ,
- or `scikit_learn_pipeline = pipeline_model.export_to_sklearn_pipeline()`

Deploy and Score

In this section you will learn how to deploy and score the model as a web service.

You can use the commands below to promote the model to space and create online deployment (web service).

Working with spaces

In this section you will specify a deployment space for organizing the assets for deploying and scoring the model. If you do not have an existing space, you can use Deployment Spaces Dashboard to create a new space, following these steps:

- Click **New Deployment Space**.
- Create an empty space.
- Select Cloud Object Storage.
- Select watsonx.ai Runtime and press **Create**.
- Copy `space_id` and paste it below.

Tip: You can also use the API to prepare the space for your work. Learn more here.

Info: Below cells are `raw` type - in order to run them, change their type to `code` and run them (no need to restart the notebook). You may need to add some additional info (see the **action** below).

Action: Assign or update space ID below.

Deployment creation

In [ ]:
```
target_space_id = input("Enter your space ID here (press enter): ")
```

In [ ]:
```
from ibm_watsonx_ai.deployment import WebService

service = WebService(
    source_instance_credentials=credentials,
    target_instance_credentials=credentials,
    source_project_id=experiment_metadata['project_id'],
    target_space_id=target_space_id
)
service.create(
    model=best_pipeline_name,
    metadata=experiment_metadata,
    deployment_name='Best_pipeline_webservice'
)
```

Use the `print` method for the deployment object to show basic information about the service:

In [ ]:
```
print(service)
```

To show all available information about the deployment, use the `.get_params()` method.

In [ ]:
```
service.get_params()
```

Scoring of webservice

You can make a scoring request by calling `score()` on the deployed pipeline.

If you want to work with the web service in an external Python application, follow these steps to retrieve the service object:

- Initialize the service by `service = WebService(target_instance_credentials=credentials,target_space_id=experiment_`
- Get deployment_id: `service.list()`
- Get webservice object: `service.get('deployment_id')`

After that you can call `service.score(score_records_df)` method. The `score()` method accepts `pandas.DataFrame` objects.

Deleting deployment

You can delete the existing deployment by calling the `service.delete()` command. To list the existing web services, use the `service.list()` method.

Running the AutoAI experiment with Python API

Info: Below cells are `raw` type - in order to run them, change their type to `code` and run them (no need to restart the notebook). You may need to add some additional info.

If you want to run the AutoAI experiment using the Python API, follow these steps. The experiment settings were generated basing on parameters set in the AutoAI UI.

```
from ibm_watsonx_ai.experiment import AutoAI

experiment = AutoAI(credentials,
project_id=experiment_metadata['project_id'])
```

```
OPTIMIZER_NAME = 'custom_name'
```

```
from ibm_watsonx_ai.helpers import DataConnection
from ibm_watsonx_ai.helpers import ContainerLocation

training_data_references = [
    DataConnection(
        data_asset_id='44785a94-5108-4bdc-974a-9481fea449e8'
    ),
]
training_result_reference = DataConnection(
    location=ContainerLocation(
        path='auto_ml/a98f60ba-cd6c-44b2-8c21-
f421bf393d86/wml_data/596e6f61-d075-4214-a0a7-
f7a6f68329a2/data/automl',
        model_location='auto_ml/a98f60ba-cd6c-44b2-8c21-
f421bf393d86/wml_data/596e6f61-d075-4214-a0a7-
f7a6f68329a2/data/automl/model.zip',
```

```
        training_status='auto_ml/a98f60ba-cd6c-44b2-8c21-
f421bf393d86/wml_data/596e6f61-d075-4214-a0a7-f7a6f68329a2/training-
status.json'
    )
)
```

The new pipeline optimizer will be created and training will be triggered.

```
pipeline_optimizer = experiment.optimizer(
    name=OPTIMIZER_NAME,
    prediction_type=experiment_metadata['prediction_type'],
    prediction_column=experiment_metadata['prediction_column'],
    scoring=experiment_metadata['scoring'],
    holdout_size=experiment_metadata['holdout_size'],
    csv_separator=experiment_metadata['csv_separator'],
    drop_duplicates=experiment_metadata['drop_duplicates'],

include_batched_ensemble_estimators=experiment_metadata['include_batched_ensem
    incremental_learning=False,
    feature_selector_mode=experiment_metadata['feature_selector_mode'],
)
```

```
pipeline_optimizer.fit(
    training_data_references=training_data_references,
    training_results_reference=training_result_reference,
)
```

Next steps

You successfully completed this notebook! You learned how to use ibm-watsonx-ai to run and explore AutoAI experiments. Check out the official AutoAI site for more samples, tutorials, documentation, how-tos, and blog posts.

Copyrights